# Neutrophil count prediction in childhood cancer patients receiving 6-mercaptopurine chemotherapy treatment

**Dalila Avdic**


A thesis submitted in partial fulfilment of the requirements of the University of Lincoln for the degree of Doctor of Philosophy

December 2018

## Abstract

Acute Lymphoblastic Leukaemia (ALL) is a common form of blood cancer, usually affecting children under 15 years of age. Chemotherapy treatment for ALL is delivered in three phases viz. induction (to achieve initial remission), intensification (to kill the majority of abnormal cells), and finally, maintenance. The maintenance phase involves oral administration of the chemotherapy drug 6-Mercaptopurine (6-MP) in varying doses to destroy any remaining abnormal cells and prevent reoccurrence. A key side effect of the treatment is a reduction in neutrophil counts that can result in a condition known as neutropenia, i.e. reduced immune system. This carries a risk of secondary infection and has been linked to 60% of ALL fatalities. Current practice aims to control neutrophil counts by varying 6-MP dosages on a weekly basis based on blood counts. However, its success is varied.

This thesis proposes a number of intelligent prediction methods to more accurately predicting neutrophil counts one week ahead using blood count data and corresponding 6-MP dosing regimens. Firstly, a well-known and robust neural network (Nonlinear Autoregressive Exogenous) is applied to blood count data to provide an initial assessment of the feasibility of such an approach. A comparative analysis of a series of more complex algorithms is then considered for more advanced, in-depth analysis viz. Multi-Layer Perceptron (MLP) and Support Vector Machines (SVM). Both methods are shown to have a prediction accuracy of around 60% on the first sample period, with the MLP also having a prediction accuracy of more than 60% in the second sample period in seven out of ten blood data points (there was 10 time-series blood data predictions). However, in comparison the accuracy of SVM is relatively low.

Finally, an incremental learning-based approach is proposed to increase the accuracy of the system and provide a realistic framework for real-time implementation. The accuracy is shown to improve considerably as more data is added, and the predicted neutrophils data is shown to follow the trend of the actual neutrophil counts.

*Ratio praeteriti scire futura facit*

## Publications Related to Research Reported in this Thesis

Avdic, Dalila and Gallimore, Michael and Riley, Mike and Bingham, Chris (2015) Neutrophil count prediction for personalized drug dosing in childhood cancer patients receiving 6-mercaptopurine chemotherapy treatment. In: International Conference on Medical and Health Sciences, June 2015, Berlin, Germany.

Avdic, Dalila and Gallimore, Michael and Bingham, Chris (2017) An Adaptive Neuro-Fuzzy Inference System Approach to Neutrophil Prediction in Childhood Leukaemia. In: The 3rd World Congress on Electrical Engineering and Computer Systems and Science, June 2017, Rome, Italy

Avdic, Dalila and Gallimore, Michael and Bingham, Chris (2018) Prediction of Neutrophil Counts for Childhood Leukaemia. Part H: Journal of Engineering in Medicine. Peer review in progress (April 2018, under review)

# Table of Contents

## Nomenclature

| | |
|---|---|
| $x$ | d-dimensional continuous data vector in Gaussian distribution |
| $\mu_k$ | d-dimensional mean vector in Gaussian distribution |
| $c_k$ | $d \times d$ covariance matrix |
| $p_k$ | Set of mixed parameters in Gaussian distribution |
| $k$ | Number if components in Gaussian distribution |
| $f_k(x\|\mu_k, c_k)$ | Probability density function in Gaussian distribution |
| $S_t$ | Markov state sequence |
| $Y_t$ | Non-Markovian output process |
| $y_i$ | Dependent variable in linear regression |
| $x_i$ | Independent variable in linear regression that $y_i$ is dependent of. |
| $k$ | Explanatory variable in linear regression |
| $\beta_0$ | Constant term in multi linear regression |
| $\beta_i$ | Coefficients relating $k$ to the variables of interest in multi linear regression |
| $e_i$ | Error term in multi linear regression |
| H | Hypothesis in ANOVA |
| $\mu$ | Population means in t-test |
| $n_1, n_2$ | Sample sizes in t-Test |
| $\overline{x_1}, \overline{x_2}$ | Means of samples in t-test |
| $s_p$ | Pooled standard deviation in |
| $s_1, s_2$ | Standard deviation of samples in t-test |
| $\bar{d}$ | Difference before and after measurements in t-test |
| $s_d$ | Standard deviation of $\bar{d}$ |

| | |
|---|---|
| $n$ | Sample number and number of nodes in Hopfield neural networks |
| $f(x)$ | Liners function in SVR |
| $w$ | Weight vector in ANN and the margin in SVR |
| $b$ | Bias value in SVR, found by calculating the average prediction error |
| $C$ | Constant parameter in SVR |
| $\xi_+, \xi_-$ | Two nonzero slack variables in both directions in SVR |
| $\varphi$ | Kernel function in SVM and activation function in NN |
| $\theta$ | Threshold in ANN |
| $x_n$ | System inputs |
| $y_n$ | System outputs |
| $H, h$ | Hidden layers in NN |
| $L$ | Set of connection links in Hopfield neural network |
| $O_{1,i}$ | Membership grade of the input nodes |
| $A_i$ | Linguistic label associated with the input node |
| $u_t$ | Input data for linear block at time $t$ in Hammerstein- Wiener Model |
| $z_t$ | Output data from the linear block at time $t$ in Hammerstein- Wiener Model |
| $\frac{B}{F}$ | Linear transfer function in in Hammerstein-Wiener Model |
| $f, h$ | Nonlinear functions in in Hammerstein- Wiener Model |
| $F_t$ | Sequence of $v \times v$ matrices in Kalman filters |
| $W_t$ | Process disturbance in Kalman filters |
| $Z_t$ | $w$ dimensional observations in Kalman filters |
| $X_t$ | $v$ dimensional state variable in Kalman filters |
| $V_t$ | Measurement noise in Kalman filters |
| $G_t$ | Sequence of $w \times v$ matrices in Kalman filters |

| | |
|---|---|
| $\delta$ | Orthogonal transformation in PCA |
| $T_m$ | Orthonormal axes in PCA |
| $\mu$ | Sample mean in PCA |
| $S$ | Eigenvectors of the covariance matrix in PCA |
| $V_p$ | Eigenvalues of $x$ in PCA |
| $c_j$ | Contributions of features to the output |
| $a$ | Arbitrary sample in PCA |
| $x_i$ | Input candidate |
| $Y$ | Regression target |
| $n$ | Number of samples when calculation the mean errors in MLP and SVR |
| $\widehat{y_i}$ | Desired value of the output when calculating the errors |
| $y_i$ | Response model when calculating the errors |
| $\bar{y}$ | Average of all model responses (average of $y_i$) |
| $\gamma$ | Gaussian kernel parameter in SVR |
| $C$ | Penalty parameter in SVR |
| $\emptyset(x)$ | Kernel function in SVR |
| $\alpha$ | Steepness of the activation function |
| $\delta$ | Local gradient in MLP |
| $N_H$ | Lower sum of neurons used in MLP |
| $N_I$ | Number of neuronal network inputs in MLP for calculating $N_H$ |
| $N_s$ | Number of training samples in MLP for calculating $N_H$ |

## List of Figures

## List of Tables

# Chapter I – Introduction

## 1.1 Introduction to leukaemia and treatment regimes

The term leukaemia refers to cancers of the white blood cells (WBC). Leukaemia usually starts in the bone marrow, the soft material in the centre of most bones, where blood cells are formed [1]. The average human adult body contains more than 5 litres of blood. Through the circulatory system, blood adapts to the body: when one is exercising, the heart pumps harder to provide more blood to the body, and therefore more oxygen. Moreover, during infections, the blood delivers immune cells to the site of infection where they aim to destroy harmful cells. However, in the case of leukaemia, a large number of abnormal cells (Lymphoblasts in Acute Lymphoblastic Leukaemia, considered in this thesis) crowd the bone marrow and flood the bloodstream, meaning they can no longer perform their proper role of protecting the body against diseases [2].

There are many types of leukeamia with some forms being more common in childern. They can be broadly categorised into two main groups [2]:

1. Acute (rapidly developing) forms, including:
    i. Acute lymphoblastic leukaemia (ALL)
    ii. Acute myeloid leukaemia (AML)
2. Chronic (slowly developing) forms

This thesis is the acute variant. Acute leukaemia means the condition progresses rapidly and aggressively, and therefore necessitates immediate treatment to facilitate a good prognosis. If the types of white blood cells affected are lymphocytes, which are mostly used to fight viral infections, the condition is called Acute Lymphoblastic Leukaemia (ALL). If the types of white cells affected are myeloid cells, which fight bacterial infections, then the condition is called Acute Myeloid Leukaemia (AML). The ALL form, in general, occurs in young children aged

between 2 and 8 years, whereas AML typically occurs before the age of 2 and during teenage years. This research focuses specifically on ALL.

The protocol of treating ALL lasts 2 years for girls and 3 years for boys and has three phases viz Induction, Intensification and Maintenance [3]. During the Induction phase the patient receives intravenous (IV) chemotherapy in an attempt to achieve initial remission. This means that leukaemia cells are no longer found in bone marrow samples; the normal marrow cells return, and the blood counts become normal. Children with standard ALL receive three drugs during this phase of treatment: chemotherapy drug L-asparagine and Vincristine and the steroid drug Dexamethasone. A fourth drug is also sometimes administered for some high-risk children, usually Daunorubicin [3]. After this phase of treatment is complete, the aim is for the patient to have less than 5% of abnormal cells, at which point there is a break in treatment until the initiation of the next phase. The Intensification phase, which usually takes between 3 and 4 months, is aimed at destroying the remaining leukaemia cells. The final Maintenance phase aims to kill any remaining cells and prevent reoccurrence.

The Maintenance phase usually lasts 18 months for girls and 30 months for boys. During this phase, the patient receives the oral chemotherapy drug 6-Mercaptopurine (6-MP), with weekly blood counts taken in order to monitor the impact of the prescribed dosage on neutrophil counts. Weekly dosages of the drug are then modified (normally 50% or 100% of the maximum dosage based on the patient's weight) in an attempt to minimise the risk of neutropenia, i.e. where the patient is considered to have no immune system.

Vincristine and a corticosteroid drug (typically Dexamethasone) are often added to the maintenance therapy. Vincristine is administered every four weeks, followed by dexamethasone for five days [4]. 6-MP is an anti-cancer oral chemotherapy drug and is taken at the same time each day [5]. The drug works by damaging the RNA or DNA that tells the cell

how to copy itself during division. 6-MP is very similar to normal substances within the cell. When cells incorporate these substances into the cellular metabolism, they are unable to divide. If cells are unable to divide, they die. The substances are called Purin Antagonism [5]. It is important to note that 6-MP acts as an immunosuppressant, supressing the production of white and red blood cells. During treatment, the aim is to achieve neutrophil counts between $1 - 1.5 \times 10^9$ neutrophils per litre of blood. A count lower than $0.5 \times 10^9$ is classed as neutropenia - no immune system. The main issue of the current dosing regime is the difficulty in accurately predicting neutrophil counts leading to frequent periods of neutropenia, and increased risk of secondary infection and periods off-treatment. The ability to predict counts more accurately is, therefore, extremely significant, leading to improved patient care and treatment outcomes, and reduced treatment costs.

This study proposes a series of intelligent system-based approaches to predict 6-MP response in ALL patients, using clinical data, in order to support clinical dosing decisions. Due to the novelty of the research, the main objective is to investigate the limitations and challenges of forecasting neutrophil counts in leukaemia (which involves the analysis of highly nonlinear time series data), and to seek ways to overcome them, leading to a realistic and robust forecasting methodology.

Specifically, this thesis proposes a number of intelligent prediction methods to more accurately predict neutrophil counts one week ahead using blood count data and corresponding 6-MP dosing regimens. Firstly, a well-known and robust neural network- Nonlinear Autoregressive Exogenous (NARX) is applied to blood count data in order to investigate the feasibility of such an approach. A comparative analysis of a series of more complex algorithms is then considered for more advanced, in-depth analyses.

Finally, an incremental learning-based approach is proposed to increase the accuracy of the system and provide a realistic framework for real-time implementation. The study first considers algorithms that have the potential of predicting neutrophil counts based on performance against a series of metrics. A small selection of these is taken forward for further analysis and development. Online learning is also considered for the development of an adaptive system in order to reduce the reliance on historical data and thereby better reflect how such an approach could be used in practice.

Ultimately it is shown that by predicting the neutrophil counts in childhood leukaemia using these machine learning techniques, a support tool for medical professionals can be provided to inform their decisions on 6-MP drug dosage.

## 1.2 Introduction to blood data constituents

Blood data has an extremely complex system of constituents all of which have a non-linear influence on the body. A clinical dataset from the maintenance phase treatment of one female ALL patient is used in this study. Data consists of multiple full blood counts and blood differentials along with corresponding 6-MP dosages. Table I shows the ranges of input variables in the dataset. Figure 1 shows the raw neutrophil counts over the time period of 588 days. For completeness, Figures 2 to 14 also show the corresponding hemoglobin, white cell count, platelets, red cell count, mean cell count, hematocrit, MCH, MCHC, lymphocytes, monocytes, eosinophils, basophils and 6-marcaptopurine during the maintenance phase over 588 days.

**Table I Input parameters in maintenance phase**

| Input | Range |
|---|---|
| Days of treatment | [1-588] |
| Hemoglobin (g/dL) | [78-169] |
| White Cell Count (10*9/L) | [0.4-11.9] |
| Platelets (10*9/L) | [87-507] |
| Red Cell Count (10*12/L) | [2.03-5.05] |
| Mean Cell Volume (fL) | [89-104] |
| Hematocrit | [0.134-0.496] |
| MCH (pg) | [28.9-34.6] |
| MCHC (g/dL) | [315-354] |
| Lymphocytes (10*9/L) | [0.25-2.83] |
| Monocytes (10*9/L) | [0-1.2] |
| Eosinophils (10*9/L) | [0-0.4] |
| Basophils (10*9/L) | [0-0.1] |
| 6-marcaptopurine (mg) | [0-80] |
| Neutrophil counts (10*9/L) | [0-8.43] |

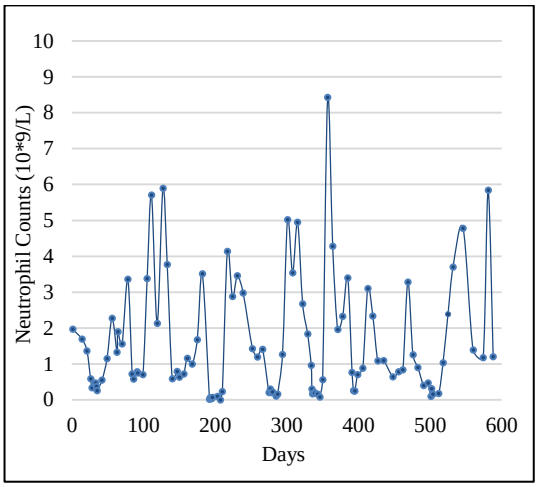



**Figure 1 Neutrophil counts during the maintenance phase**

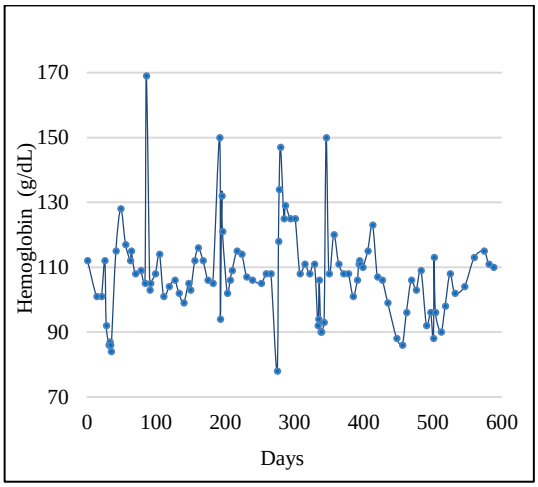



**Figure 2 Hemoglobin during the maintenance phase**

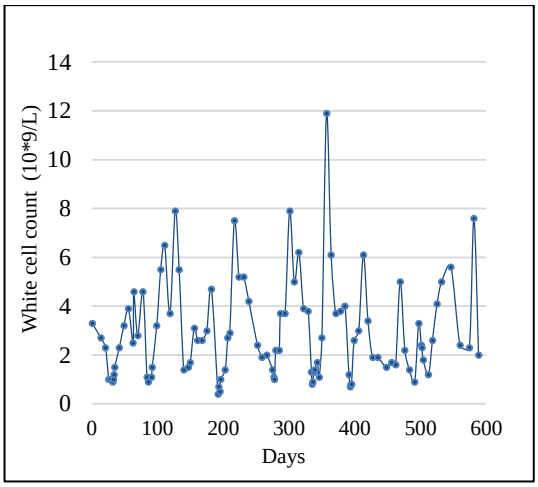



**Figure 3 White cell counts during the maintenance phase**

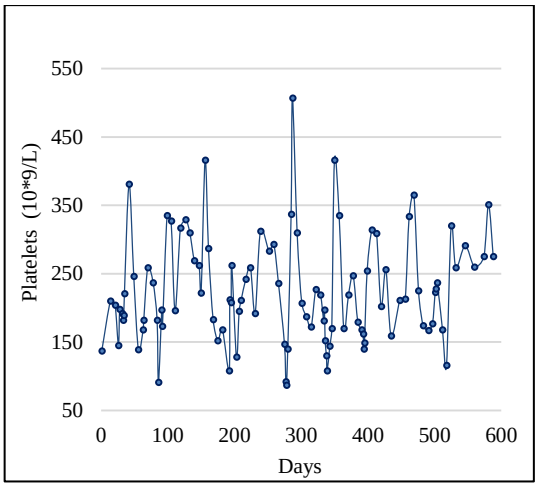



**Figure 4 Platelets during maintenance phase**

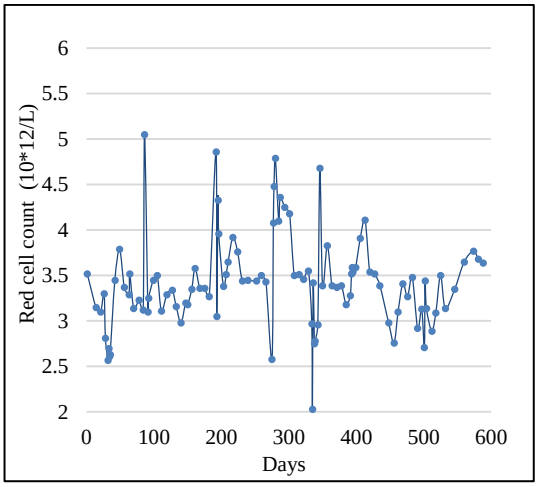



**Figure 5 Red cell counts during the maintenance phase**

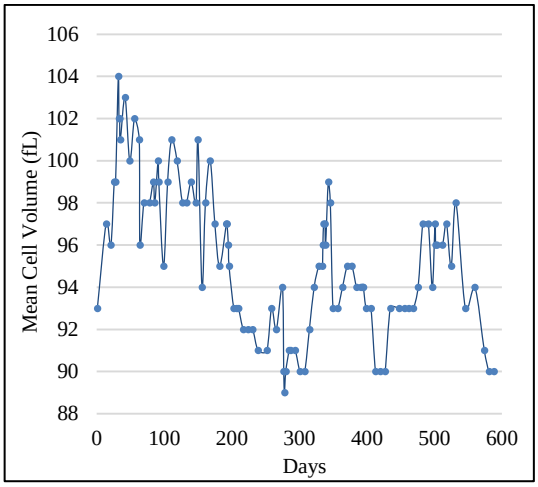



**Figure 6 Mean Cell Volume during the maintenance phase**

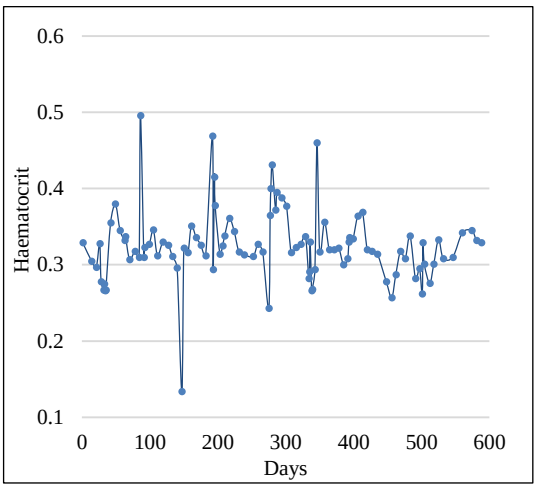



**Figure 7 Haematocrit during the maintenance phase**

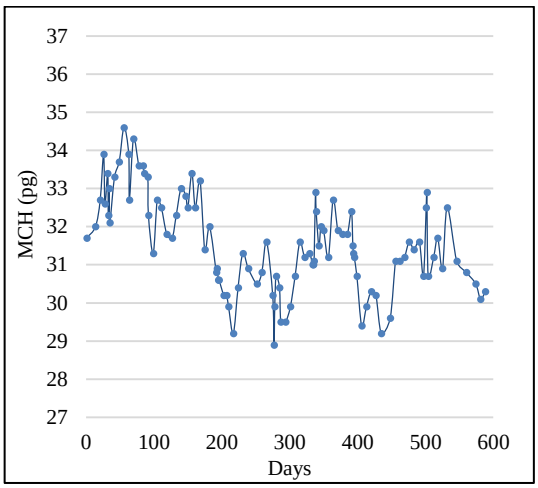



**Figure 8 MCH during the maintenance phase**

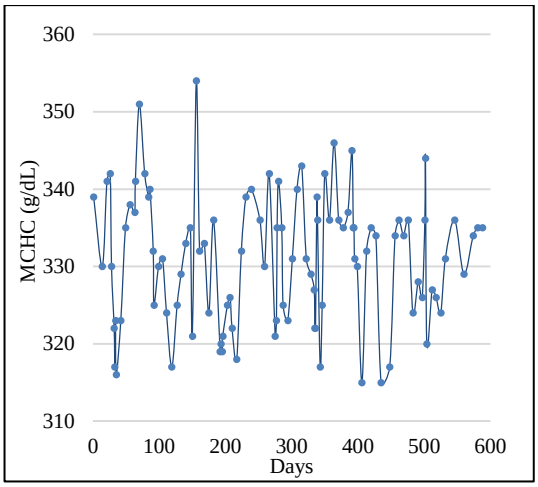



**Figure 9 MCHC during the maintenance phase**

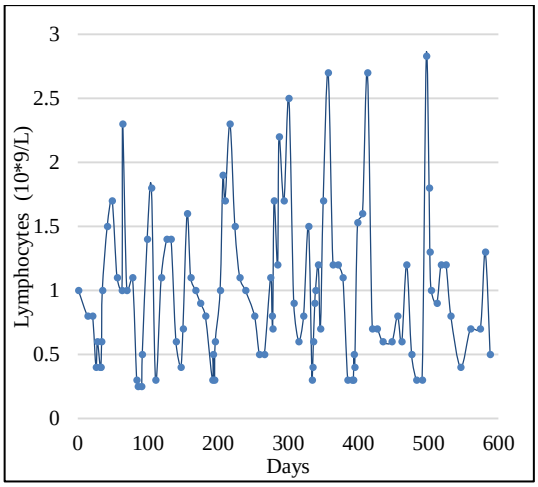



**Figure 10 Lymphocytes during maintenance phase**

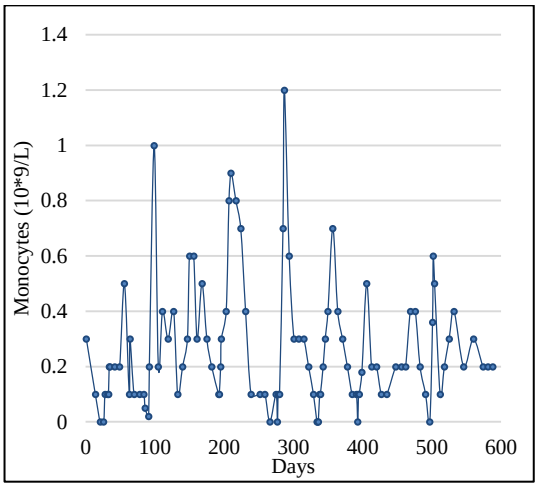



**Figure 11 Monocytes during maintenance phase**

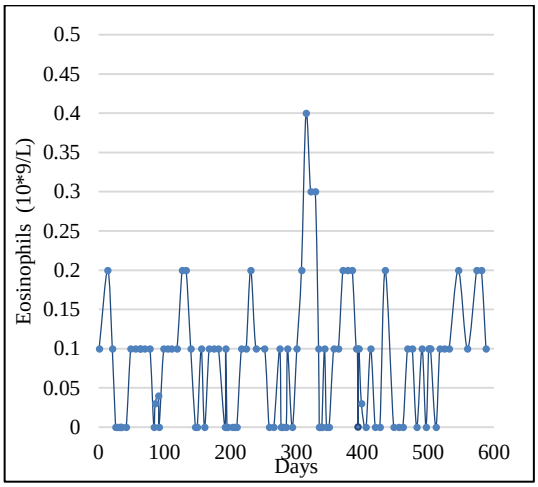



**Figure 12 Eosinophils during maintenance phase**

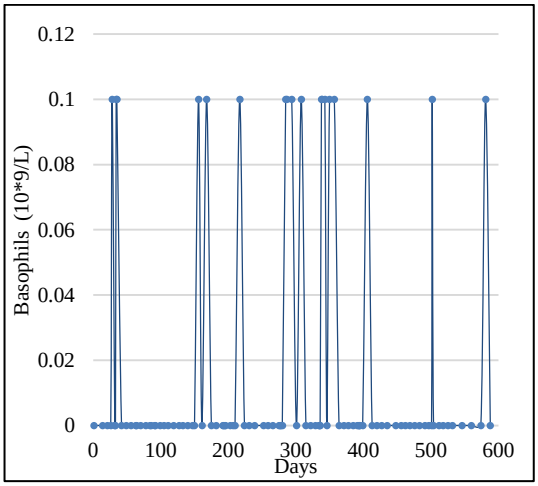



**Figure 13 Basophils during maintenance phase**

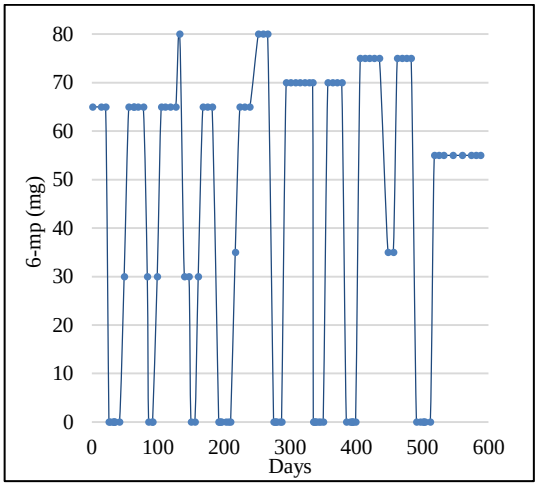



**Figure 14 6-marcaptopurine during maintenance phase**

## 1.3 Study objectives

The main objective of this research is to develop and apply algorithm or algorithms to blood data to predict neutrophil counts at least one week ahead in order to support the clinicians decision of how much 6-marcoptopurine drug to give patience. With the purpose of achieving this objective, smaller objectives are proposed, which are:

- Data pre-processing
- Proposal of the methodology for selecting the input parameters for short term neutrophil count prediction
- Comparison of different algorithms appropriate for this data kind
- Analysis and further development of algorithms for solving the problem of short-term prediction
- Development and application of modified algorithms for short-term online prediction

## 1.4 Thesis organization

The thesis is organised as follows:

***Chapter 1*** explains the motivation behind the study and the overall objectives of the research. The problem is defined along with an introduction to the data set used throughout the study.

***Chapter 2*** presents a broad review of machine learning techniques that are identified in literature as being used in similar research problems.

***Chapter 3*** provides an overview of the patient's data used in the analysis, along with selected pre-processing techniques viz. the normalization and interpolation of data. This chapter also

looks at identifying the most important blood constituents to provide neutrophil predictions using principal component analysis (PCA).

***Chapter 4*** presents an initial feasibility study on neutrophil prediction (one-day ahead) using a NARX neural network structure.

***Chapter 5*** identifies other techniques that may be suitable for this kind of prediction problem and provides a relative appraisal of each. Following the appraisal, the most relevant techniques identified for this specific application are selected for further investigation. Specifically, SVM and MLP, which are both then applied and appraised.

In ***Chapter 6*** more advanced multi-step ahead neutrophil prediction methodologies are considered viz. the Support Vector Machines (SVMs), and subsequently in ***Chapter 7***, the Multi-Layer Perceptron (MLP).

***Chapter 8*** extends the work of Chapter 7 and proposes an on-line incremental learning methodology based on the MLP algorithm to increase prediction accuracy and provide a realistic framework for real-time implementation.

***Chapter 9*** presents conclusions and further work.

# Chapter II – Review of Relevant Machine Learning Methods

## 2.1 Introduction

The problems addressed in this thesis lie at the intersection of machine learning and time series forecasting. Machine learning is relatively fledgling field that has evolved in the science community and is so-termed to distinguish it from human learning. Learning from data consists of using a set of observations to expose an underlying process. Different types of learning have been considered in the literature depending on the type of data involved in the learning process. Examples include supervised learning, unsupervised learning and online learning methods. Alternative examples involve regression, clustering and classification algorithms.

## 2.2 Time series prediction

Time series modelling is a dynamic research field that has attracted considerable attention from the researcher community over the past few decades. The main aim is to carefully collect and rigorously study past observations of a time series to develop an appropriate model that describes the inherent underpinning structure of the series. The resulting model is then used to generate future predictions expected from the series. Different types of learning techniques have been considered in the literature include statistical and probabilistic methods, neural network-based methods and other regression models. A summary of relevant machine learning methods is given in Figure 15, which is used as a seed for further discussion.

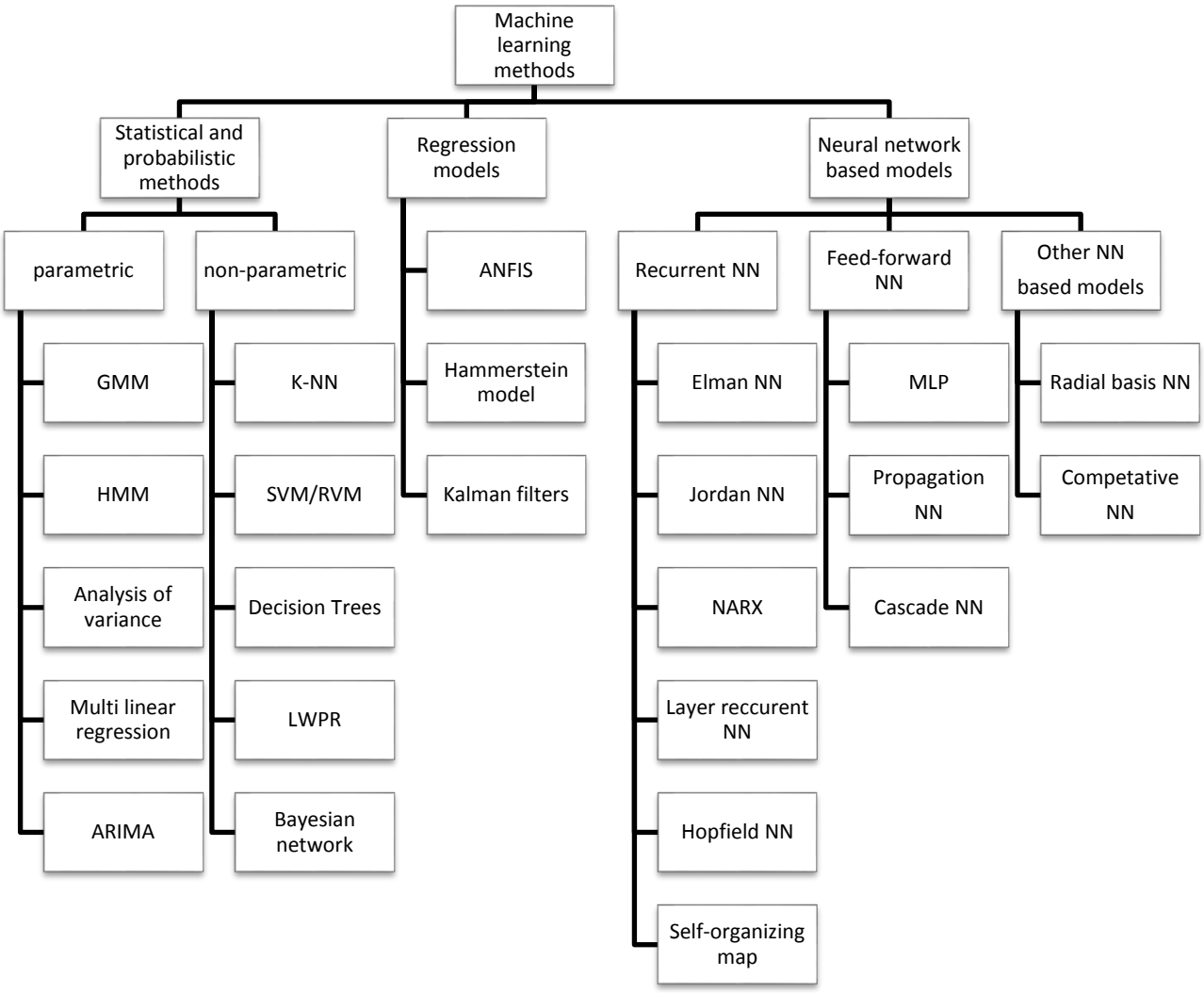



**Figure 15 Candidate machine learning methods**

## 2.3 Statistical and probabilistic parametric methods for prediction

Parametric methods are a classification of statistical methods based on prior assumptions about the distribution of the underlying population from which the sample was taken, for instance, whether the data are normally distributed. A primary disadvantage of all such techniques is they can be insensitive to small changes in system behaviour, thereby leading to inaccurate results [6] [7]. The two most commonly used methods are the Gaussian Mixture Model (GMM) and Hidden Markov Model (HMM). Although these are not directly applicable for the type of Multi-

Input-Single-Output (MISO) system consider in this thesis, an appraisal of their features is given below to highlight the relevance of other methods.

### 2.3.1 Gaussian Mixture Model

The Gaussian Mixture Model (GMM) is a statistical parametric probability density function represented as a weighted sum of component densities called Gaussians. Gaussian mixture models are used in pattern recognition problems [8], prediction [9], monitoring [10], segmentation [11], discrimination [12], clustering [13], fusion [14] [15] and supervised learning of multimedia data [16]. GMM parameters are usually estimated using a training process based on the data series using the Expectation-maximization (EM) algorithm [17]. However, EM algorithms are not used when the number of data attributes or the dimensions of the vectors is large since the parameter accuracy of the model deteriorates.

The structure of the Gaussian model can be represented by [18]:

$$g(x|\lambda) = \sum_{k=1}^{m} p_k \, f_k(x|\mu_k, c_k) = L(\lambda|x) \qquad (1)$$

where $x$ is a D-dimensional continuous data vector, $p_k$ are the mixture weights and $f_k(x|\mu_k, c_k)$ are the compound Gaussian densities. $L(\lambda|x)$ is considered as the likelihood function and the aim is to estimate a set of parameters $\lambda$ such that $L$ is maximised. Each compound density is a D-variate Gaussian function of the form:

$$f_k(x|\mu_k, c_k) = \frac{1}{(2\pi)^{\frac{d}{2}}|c_k|^{\frac{1}{2}}} exp\left\{-\frac{1}{2}(x-\mu_k)^T {c_k}^{-1}(x-\mu_k)\right\} \qquad (2)$$

Equation (2) has $d$-dimensional mean vector $\mu_k$, and the $d \times d$ covariance matrix $c_k$ is called the component density function CFD of the GMM and $p_k$ is a set of mixing parameters that satisfy the following condition:

$$p_k \geq 0, \sum_{k=1}^{n} p_k = 1 \quad (3)$$

When using high dimensional vectors, there is usually insufficient data for accurate estimation of the parameters of the covariance matrix. This has been the focus of much GMM related research over recent years and many precise parameter estimation methods have been proposed that can improve their accuracy; see for example [19] and [20]. Others have also proposed methods for reducing the number of parameters by regarding the small eigenvalues of the sample covariance matrix as constant [21].

GMM are typically useful when trying to determine natural groupings in data, whilst their value for input-output system modelling is limited.

### 2.3.2 Hidden Markov Model

The Hidden Markov Model (HMM) is a tool for modelling time series data and has the type of structure shown in Figure 16. HMM is a parametric statistical model following a Markov process but with "hidden" states. HMMs are a generalization of Markov chains in which the underlying Markov state sequence, $S_t$, is observed through a channel, leading usually to a non-Markovian output process, $Y_t$. They have been used in many applications of temporal pattern recognition from the mid-1970s, with the first applications being in speech recognition [22] [23] [24] [25], and then bioinformatics [26] [27] [28], data compression [29], and artificial intelligence [30] [31]. Figure 16 shows a typical HMM layout.

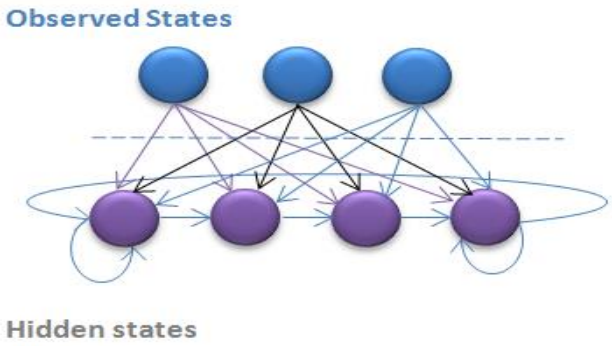



**Figure 16 Hidden Markov Model**

HMMs have two main properties [32]:

- They assume that the observation at time $t$ is generated by a process whose state, $S_t$, is hidden from the observer
- They assume that the state of its hidden process is such that, when given a value of $S_{t-1}$, the current state $S_t$ is independent of all the states prior to *t-1*. This is a first-order Markov property and that $n^{th}$ order Markov processes are where $S_t$ is given by $S_{t-1} \dots S_{t-n}$ and it is independent for $S_\tau$ for $\tau < t - n$.

When taken together, these Markov properties mean that the joint distribution of a sequence of states and observations can be factored in the following manner [32]:

$$P(S_{1:T}, Y_{1:T}) = P(S_1)P(Y_1|S_1)\prod_{t=2}^{T} P(S_t|S_{t-1})P(Y_t|S_t) \quad (4)$$

HMMs aim to model systems that are based on these two assumptions, 1: Markov model chain emits a sequence of observable outputs and, 2: the latest output depends only on the current state of the system, and allow (amongst other things) the following [25]:

- Conclusion of the most likely sequence of states that produce a given output sequence.
- Calculation of the most likely next state, i.e. predicting the next output.
- The ability to calculate the probability that a given sequence of outputs is created from the system.

A variable can only be predicted using this model if there exists some information about the present and everything used in the past is unimportant. The term 'hidden' refers to the fact that states are not directly observable, but that the output is dependent on the state values [33].

### 2.3.3 Multiple linear regression

Multiple linear regression is the most common form of linear regression analysis. As a predictive analysis technique, multiple linear regression is used to fit a single line through a scatter plot, as shown in the example of Figure 17.

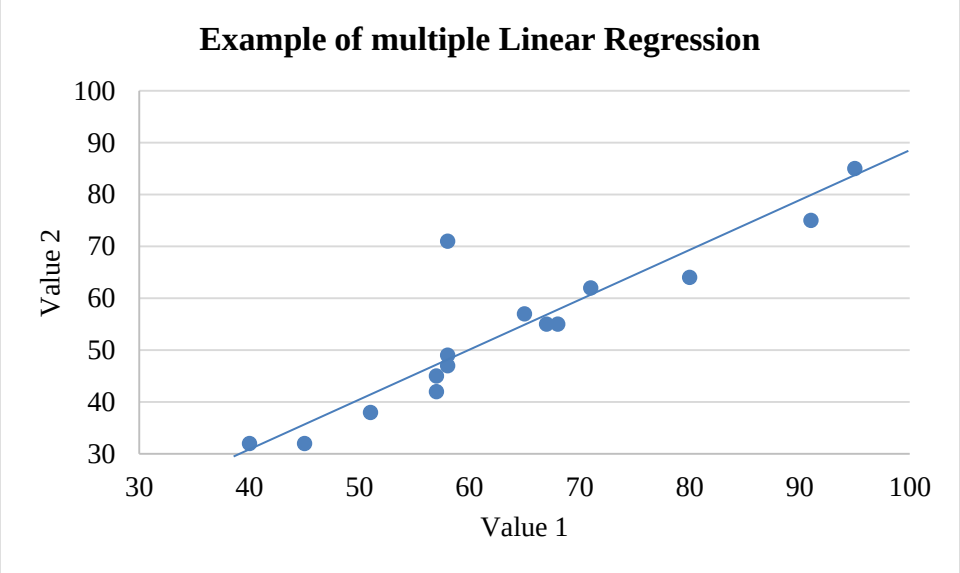



**Figure 17 Example of Multiple Linear Regression**

When using multiple linear regression, there exist $k$ explanatory variables, and the relationship between the dependent variable and the explanatory variables is represented by the following equation:

$$y_i = \beta_0 + x_{i1}\beta_1 + x_{ik}\beta_k + e_i \qquad i = 1,2,\dots,n \tag{5}$$

where: $\beta_0$ is the constant term and $\beta_1$ to $\beta_k$ are the coefficients relating the $k$ explanatory variables to the variables of interest. In (5), $y_i$ is considered as the dependent variable whose value depends on the covariates $x_i$. Parameters $\beta_i$ are unknown and are typically variances to be estimated from the data. Error terms are normally distributed and denoted as $e_i$ [34].

A multiple linear regression model is underpinned by five basic assumptions concerning the way in which the data are generated [35]:

1. The dependent variable can be calculated as a linear function of the covariates, plus an error term. Thus, it should have the form of Equation 5.

Issues about this assumption:

• Wrong regressors - absence of relevant covariates and presence of irrelevant covariates.

• Nonlinearity - the relationship between the dependent variable and the covariates is not linear.

2. Expected value of the error term is zero. An estimator with the expected value of zero is called unbiased.

3. The error terms all have the same variance and are not correlated with one another.

Issues about this assumption:

• Heteroscedasticity - the error terms do not have the same variance.

4. Covariates can be considered fixed in repeated samples, which means it is possible to redraw the sample with the same values for the covariates.

Issues about this assumption:

• Errors in variables - errors in measuring the covariates.

• Auto regression - using a lagged value of the dependent variable as a covariate.

5. The number of dependent variables is greater than the number of covariates and that there is no exact linear relationship between the covariates.

Issues about this assumption:

• Multicollinearity - two or more covariates are approximately linearly correlated in the sample data.

### 2.3.4 ARIMA

Among common time series models, a fundamental variant is the autoregressive integrated moving average (ARIMA) model (Hamilton 1994) [36], developed from the autoregressive model (AR) and the moving average model (MA). Theoretically, if there is no missing data (Weigend 1994) [37] for a stationary time series, then this model can learn an identified underlying process to mimic observations for predicting future data points. In practice, ARMA can describe the behaviour of a noisy linear dynamical system and is able to represent several different types of time series due to its flexible modelling capability.

Despite its great value and successful application, ARMA assumes the underlying model is linear, which hinders its applications to many challenging real-world time series problems.

Moreover, when their relative performance is compared in literature with models such as neural networks, neural networks almost always outperform them. For instance, Yao et al. [38] compared stock forecasting performance of ANN and ARIMA models and showed that the ANN model obtained better returns than the conventional ARIMA. Similarly, Hansen et al. [39] compared the prediction performance of ANNs and ARIMA on time series prediction to show that the ANNs outperformed ARIMA in predicting stock movement direction, as the former was able to detect hidden patterns.

### 2.3.5 Hypothesis Testing

T-test and Analysis of Variance (ANOVA) are two parametric statistical techniques used to test a hypothesis. They are based on common assumptions that: the population from which sample is drawn is normally distributed, homogeneity of variance, random sampling of data, independence of observations.

**Analysis of variances (ANOVA)** is a parametric statistical modelling method developed by R. A. Fisher [40] in the 1920's and 1930's and is therefore also known as Fisher's analysis of variances or Fishers ANOVA. Analysis of variance is used when comparing means between three or more separate independent groups. There are a few variations of ANOVA that can be used to test differences between samples:

1. One-way ANOVA
2. Two-way ANOVA
3. Three-way ANOVA
4. MANOVA

However, before exploring the details of ANOVA, it is first necessary to understand the differences between a Factor and Factor Levels. Factor is the characteristics under consideration

that are believed to influence the measured observation. Factor levels are values of the vector. If the calculation uses one-way ANOVA, it means that the sample only has a single factor. Conversely, when two-way ANOVA is used, a second factor is added, and so on. A MANOVA (Multivariate analysis of variances) is also a type of ANOVA, where one is able to analyse two or more related dependent variables. The first three ANOVA described test the difference in means, whereas MANOVA tests the difference of two or more vectors in means.

Considering one-way (factor) ANOVA the following assumptions are made [41]:

- Independence: the observations are obtained independently and randomly from the populations defined by the factor levels.
- Normality: the population at each factor level is normally distributed.
- Homogeneity of variances: these normal populations of factor level have a common variance, $\sigma^2$.

A null hypothesis and an alternative hypothesis are also required, which assume there will be no differences between groups that are tested (therefore no significant results will be revealed) and there will be a difference between groups as indicated by the ANOVA on the data collected, respectively. Hypotheses that are tested are [41]:

$H_0: \mu_i = \mu$ All $i = 1,2, \dots, k$

$H_1: \mu_i \neq \mu$ Some $i = 1,2, \dots, k$

where $\mu_i$ is the population mean for level $i$.

Hypothesis is important for understanding ANOVA. For example, if two groups of 50 people are considered, with the aim of identifying a particular difference between the two groups, i.e. which group performs better in different circumstances, one can simply summarize the data collected, compare the performance achieved and conclude on that basis. However, this has

limited appeal as the result only applies to the particular people in the groups. If another set of a different 50 individuals, the study may show different results. Therefore, in Hypothesis testing, the people in the study are samples and the large group they represent is the population. Using this technique, it can be established whether a conclusion extends beyond the samples into the wider population.

ANOVA has been used in many applications. For instance, the social sciences use ANOVA to identify what factors influence peoples' opinions and behaviours. Analysis of variances has also been used in drug prediction (albeit rarely), for example, in machine learning prediction of cancer cell sensitivity to drugs based on genomic and chemical properties by M. P. Menden *et al.* [42]. ANOVA has also been applied in the identification of drug-to-oncogene associations [43].

ANOVA is an efficient method for analysing experimental data. One-way ANOVA is characterised by simple calculations. However, two-way ANOVA is more advantageous as it reduces random variables. In research, using a two-variable design offers many advantages over using a one-variable design. The first advantage is increased efficiency. Another advantage is that the interaction of two variables in the design can be analysed. However, when comparing ANOVA with T-test (discussed next) it is found to be generally more applicable. For instance, if two data means are compared, ANOVA will offer the same results as the subsequently discussed T-test. ANOVA is a method to compare two or more means and therefore it simplifies T-test to more than two groups. ANOVA compares all means simultaneously and maintains the type 1 error probability at the designated level. Type 1 errors occur when the null hypothesis ($H_0$) is true but it is rejected.

Although ANOVA is an effective method for analysis of experimental data, it can also be complex as it has got many varieties that can apply in different experimental contexts.

Therefore, a 'wrong type' of ANOVA can inadvertently be applied (especially when dealing with large data sets) leading to erroneous conclusions from an experiment. Another disadvantage of this technique is that when it determines the differences between the groups it is not clear which one is considered the variant.

**T-test** is a parametric statistical technique used when comparing means between two distinct/independent groups. The technique identifies differences between two groups based on some variable of interest, and it is used when comparing two quantitative measurements taken from the same individual. For instance, it can be used to compare the average shopping time of women with that of the average shopping time for men over a year, when considered as a continuous variable. T-test uses the t-statistic (test statistic), t-distribution and the degree of freedom to determine a probability value ($p$) that is used to describe whether the population means differ [44]. As with ANOVA, it firstly defines the null hypothesis and an alternative hypothesis which correspond to the underlying question of interest. There are two types of t-test groups: Independent Sample T-test (information of one sample does not provide any details about the second sample) and Paired Sample T-test (information of one sample gives some details about the other sample). The evaluation of both types is shown below:

**Independent Sample T-test**

The steps for this test are:

- Find the hypostases $H_0$ and $H_1$ with using the Greek letter $\mu$ to refer in differences in population
- Find the standard error
- Calculate the t-test statistic with Equations (6) and (7):

$$t = \frac{\overline{x_1} - \overline{x_2}}{\sqrt{s_p^2\left(\frac{1}{n_1} + \frac{1}{n_2}\right)}} \quad (6)$$

$$s_p = \sqrt{\frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2}} \quad (7)$$

where

- $\overline{x_1} - \overline{x_2}$ is the difference of the mean
- $n_1$ and $n_2$ are the sample sizes
- $s_1$ and $s_2$ are the standard deviations of sample
- $s_p$ is the pooled standard deviation

The assumptions that need to be met to ensure test validity are [45]:

- One variable is continuous and the other one is dichotomous
- Two distributions have equal variances
- Observations are independent
- Two distributions are normally distributed

**Paired Sample T- test**

The difference in analysing Paired-Sample T-test from Independent Sample T-test is that this T-test tests the null hypothesis that the mean difference between the before and after test score, is zero. The pair t-test statistic is calculated using:

$$t = \frac{\bar{d}}{s_d/\sqrt{n}} \quad (8)$$

where $d$ is the difference between before and after measurments, and $s_d$ is the standard deviation of those differences. Considering the t-test, the assumption of homogeneous variances does not apply as it is based only on one variable, $d$. This test is not recommended since undertaking multiple two sample t- tests results in an increased chance of committing a type 1 error, as explained previously.

## 2.4 Statistical and probabilistic non-parametric methods

E. Walsh [6] states "Statistical procedure is of nonparametric type if it has properties which are satisfied to a reasonable approximation when some assumptions that are at least of a moderately general nature hold." Non-parametric is a classification of statistical procedures that do not rely on assumptions about the shape and parameters. The advantage of non-parametric tests is that they make fewer assumptions about the distribution of measurements in the data. However, there two main disadvantages:

- Less statistically powerful: there is a smaller probability that the procedure will tell us that two variables are associated with each other when they in fact are. The procedure requires a relatively large sample size to have the same performance as a parametric test.
- Results are not readily interpolated as they usually use rankings of the values in the data and not the actual data—with knowledge that the difference in mean ranks between two groups is 5 does not really assist in understanding the characteristics of the data. However, knowing that the neutrophil count of patients taking a drug was 0.2 lower than the mean neutrophil count from previous week/s of patients on a different treatment, is both intuitive and useful [6] [7].

### 2.4.1 k-Nearest Neighbour

K-Nearest Neighbour (k-NN) is a nonlinear, non-parametric forecasting method. It can be used for both classification and regression predictive problems.

It is based on the concept that pieces of time series in the past resemble patterns of the future. The algorithm locates such patterns as “nearest neighbours”, typically using the Euclidean distance.

k-NN is a relatively simple algorithm based on feature similarity. How closely out-of-sample features resemble the training set determines how a given data point is classified. Figure 18 shows an example of a k-NN classification problem. The test sample (green circle) should be classified either to the first class of blue triangles or to the second class of red squares. If $k = 3$ it is assigned to the second class as there are 2 squares and only 1 triangle inside the inner circle. If $k = 5$ it is assigned to the first class as there are 5 blue triangles and only 4 red squares inside the circle.

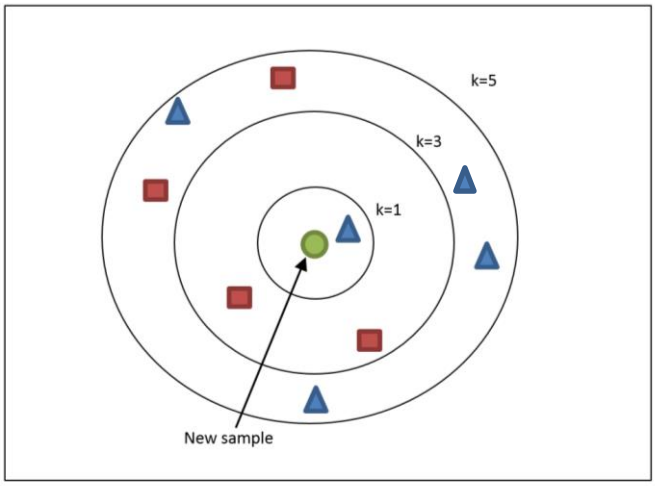



**Figure 18 Example of k-NN in classification**

The k-NN algorithm more often used in hybrid algorithms. However, models based on k-NN are widely used. As previously discussed, a major attraction of the nearest neighbour procedure is its relative simplicity. For realisation, they require only a measure of distance in the sample space, along with samples of training data, hence their popularity as a starting point for refinement, improvement and adaptation.

The basic algorithm was extended by Y. Qin *et al.* [46] for improving the processing of complex nonlinear data sets with a novel clustering method based on a hybrid k-NN which shows a significant improvement in performance over existing nonlinear clustering methods.

The algorithm's advantages are that no prior assumptions are required about the characteristics of the model and complex concepts can be readily learned using relatively simple procedures.

### 2.4.2 SVM

Support Vector machines (SVM) was introduced in 1979 by Vapnik and subsequently used in 1995 for regression and classification [47]. SVMs are a supervised machine learning method where training data is mapped and separated into two classes by a hyperplane calculated from the results of quadratic programming. The hyperplane is then re-mapped back into the input space allowing a non-linear separation of the input data. However, the main drawback of this methodology is that it is only applicable to two-class problems.

SVM's have mainly been applied to classification and regression problems, termed respectively SVC (Support Vector Clustering) and SVR (Support Vector Regression). SVM has become popular for pattern recognition problems, particularly for binary classification. SVMs are useful in the analysis of many complicated real-life problems with relatively small data sets [48] [49]. The objective is to find the state in which the distance between the two classifications is maximum [50] and thereby find a hypothesis that minimizes the experimental error [47]. Minimizing the error is equivalent to finding the hyperplane that is at the maximum distance from the closest training sample for the two classes, as shown by S. Bernhard *et al.* [51]. Classification can be achieved by a linear or non-linear separating surface in the input space. SVM for classification is illustrated in Figure 19.

Support vectors are the data points from the $+ve$ and $-ve$ classes that are closest to the separating hyperplane. For instance, in Figure 19 many hyperplanes can divide the red and blue classes into two, but only one hyperplane uniquely identifies with the support vectors. This hyperplane passes midway through the support vectors. A support vector is to be termed a $+ve$ support vector if it is from the positive class, otherwise it is termed a $-ve$ support vector. Hyperplanes that pass through the support vectors are called margin lines. There is two margin lines, one that passes through the $-ve$ support vector and one that passes through the $+ve$ support vector. The distance between the two is called the margin. In Figure 19, a two-dimensional graphical illustration of support vectors, margin lines and the margin is shown.

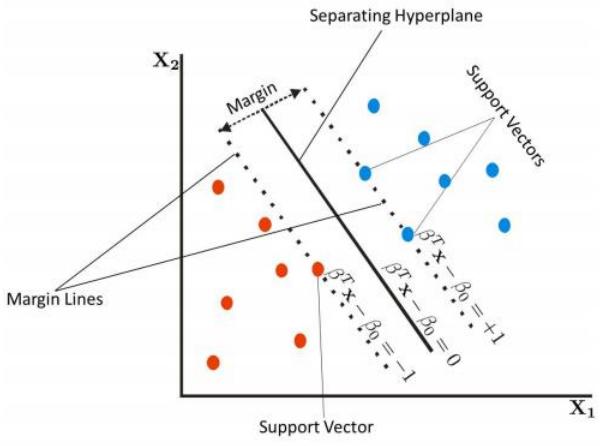



**Figure 19 Graphical illustration of Support Vectors and Margin for SVM**

Although SVMs have mainly been used for classification purposes, their underlying principles can be extended to regression and time series prediction, as demonstrated by Thissen *et al.* [52]. Support Vector Regression (SVR) uses similar principles as the SVM for classification with only a few minor differences. In the case of regression, a margin of tolerance ($\varepsilon$) is set and the objective is to minimize the error, identifying the hyperplane that maximizes the margin.

**SVR** uses the $\varepsilon$-insensitive loss function as depicted graphically in Figure 20 [53].

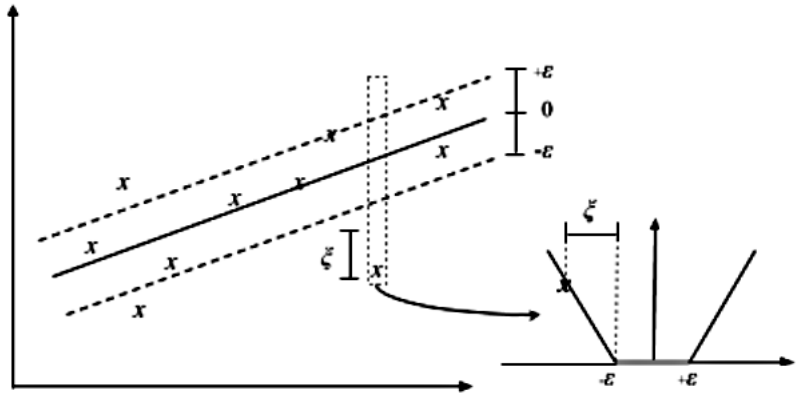



**Figure 20 $\varepsilon$-insensitive loss function for a SVR [53]**

An advantage of using this function is that it can provide a tolerance against noise. SVR approximates a linear function *f(x)* with the following form:

$$f(x) = w^T x + b \tag{9}$$

where the coefficients *w* and *b* are weight vector and bias terms, respectively. Constraints can be placed on the function to form the following optimisation problem:

$$\min \frac{1}{2}||w||^2 + C\sum \xi_+ + \xi_- \tag{10}$$

where $\xi +$, $\xi -$ are the two nonzero slack variables. The bounded area aims at fitting the data with an admissible parameter $\varepsilon$. The constant parameter, $C > 0$, allows a trade-off between complexity (flatness) of the function and the amount up to which deviations larger than $\varepsilon$ are tolerated. Data samples outside of the bounded zone within the distance of slack variables are the support vectors.

In recent years extensive research has been undertaken to extend and apply SVM for time series forecasting. As a result, many different SVM forecasting algorithms have been derived, for

example, the Critical Support Vector Machine (CSVM) [54] algorithm, and the Least Square Support Vector Machine (LS-SVM) [55] [56] algorithm.

Least Square Support Vector Machines (LSSVM) is a least square version of SVM, Suykens, *et al* [57], and is a further adaptation of Vapnik's [47] original SVM formulation for solving pattern recognition problems. LS-SVM is a quadratic programming technique with an infinite number of unknown parameters and constraints. However, the constraints can be solved using Lagrange multipliers by transforming the primal space into a dual space, and therefore the problem of having an infinite number of unknown parameters can be avoided. During the transformation, the Mercer's condition for mapping low-dimensional space to high-dimensional space is applied [58]. The advantage of LS-SVMs over SVMs is not only that they solve an optimization problem using equality constraints rather than inequity constraints, but that it is easier to design a modified version of LS-SVM due to its simpler formulation; for instance, as with the weighted LS-SVM for improving the robustness in nonlinear function estimation by Suykens *et al*. [59] in 2002. Furthermore, the SVM is only used for static problems, where LS-SVM can be applied to both static and dynamic problems.

RVM (shown in Figure 21) is identical in form to SVR. However, by reforming the support vector solution with 'expectation maximisation learning', the Relevance Vector Machine is created. A Bayesian framework is used for RVM, thereby providing posterior probabilistic outputs, that typically provide much sparser solutions than SVR; both of which are desirable qualities (as described in [60]).

The RVM is a special case of a sparse linear model, where the basis functions are formed by a kernel function $\phi$ centred at the different training points [61]:

$$f(x) = \sum_{i=1}^{N} w_i \phi(x - x_i) \qquad (11)$$

While this model is similar in form to the support vector machines (SVM), the kernel function does not need to satisfy Mercer's condition, which requires $\phi$ to be a continuous symmetric kernel of a positive integral operator.

The benefit of using RVM over SVM is that there is no requirement to determine any parameters after training. However, this benefit also has a penalty – namely, that the training procedure involves the solution of a non-convex optimisation problem. When training an RVM (and neural networks), it is therefore possible to converge on local minima.

Compared to SVM, RVM is found to be advantageous in several aspects, including [61]:

1) The number of relevance vectors can be much smaller than that of support vectors;

2) RVM does not require tuning of a regularization parameter ($C$) as in SVM during the training phase. A drawback, however, is that the training phase of RVM typically involves a highly nonlinear optimization process.

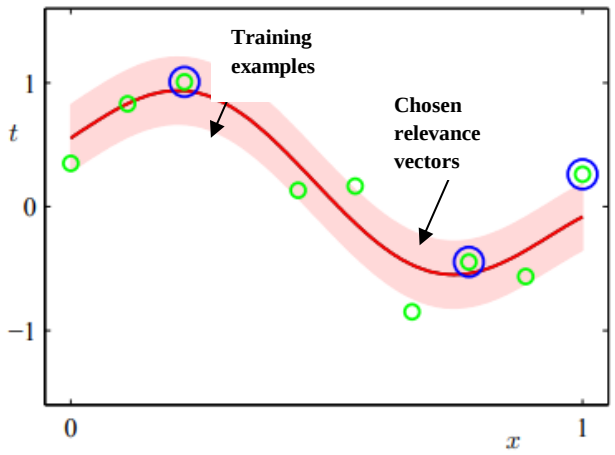



**Figure 21 Example of relevance vector machine**

### 2.4.3 Decision Trees

Decision trees for regression or classification problems in terms of a tree structure. They separate the data into smaller parts. A simple decision tree is shown in Figure 22.

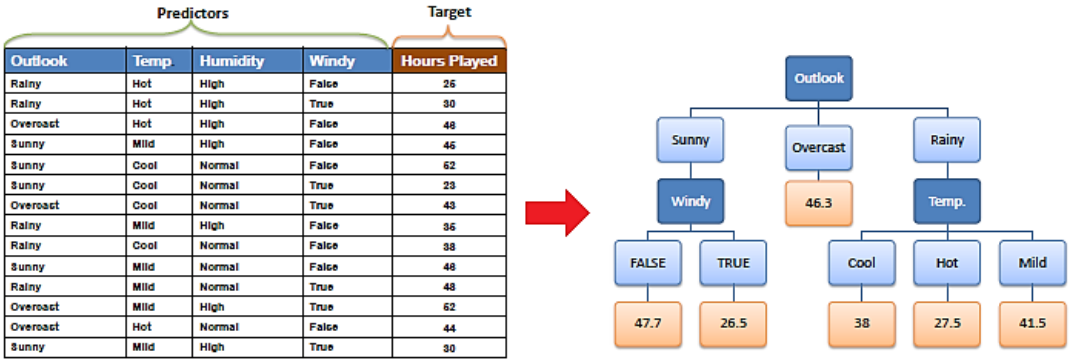



**Figure 22 Simple decision tree example [62]**

In the above example, a decision node (outlook) has 3 branches (sunny, overcast, and rainy) each representing values for the attribute tested. Leaf node (hours played) represents a decision on the numerical target.

Decision Trees typically incur low memory usage and are readily interpreted. They also provide a rapid methodology for fitting and prediction, and can be applied to categorical prediction problems. Their main drawback is that they often provide prediction of lower accuracy compared to alternative algorithms [63].

### 2.4.4 Bayesian networks

Bayesian networks are an example of a non-parametric graphical model (GM) for representing conditional independencies between sets of random variables. A simple Bayesian network is shown in Figure 23.

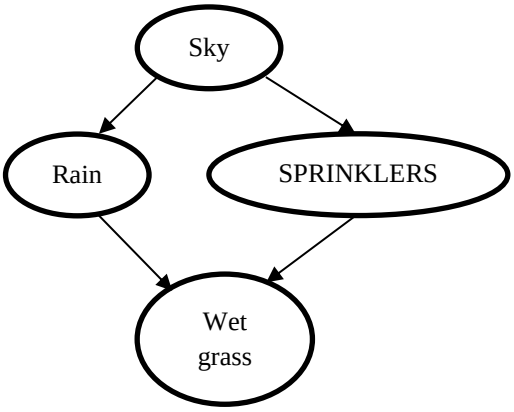



**Figure 23 A simple Bayesian network [64]**

In Figure 23, the condition of the sky (whether it's rainy or sunny) will influence whether the sprinklers are on or off. And both rain and sprinklers influence whether the grass is wet. In a Bayesian network, a connection between A and B can be informally interpreted as A "causes" B [64].

Early research on the use of Bayesian networks for prediction includes dynamic network models [13], with, for example, a clinical application to predicting sleep apnoea [14]. There is also a reported application of Bayesian networks for clinical time series analysis, where data is analysed from chronic obstructive pulmonary disease (COPD) patients to construct a model to predict the occurrence of exacerbation events, such as episodes of decreased pulmonary health status. However, these networks are more suitable for multi-input-multi-output systems and are not appropriate for the non-linear characteristics that blood constituents have on the body, as studies here.

## 2.5 Neural Network-based models for prediction

### 2.5.1 Introduction

Artificial Neural Networks (ANNs) consist of simple processing elements-neurons whose design is motivated by the structure of the human brain [65]. The application of ANN's have

become increasingly popular in recent years for analysing complex problems, including those in finance, medicine, physics and engineering. There is no standardised definition of a neural network, but those most recognized are:

- "A neural network is a system composed of many simple processing elements operating in parallel whose function is determined by network structure, connection strengths and the processing performed at computing elements or nodes"- By DARPA Neural Networks Study (1988) [66]
- "A neural network is a massively parallel distributed processor that has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects:

    1. Knowledge is acquired by the network through a learning process.

    2. Interneuron connection strengths known as synaptic weights are used to store the knowledge", Haykin (1994) [67]

### 2.5.2 From Biological to Artificial Neurons

The capability of the brain to solve complex nontrivial problems that remain impossible even using the most recent computer technology. Recognition of the brain's impressive capabilities has led to increasing interest in the development of ANNs [68] based on a basic model of a brain neuron, as shown in Figure 24 [69].

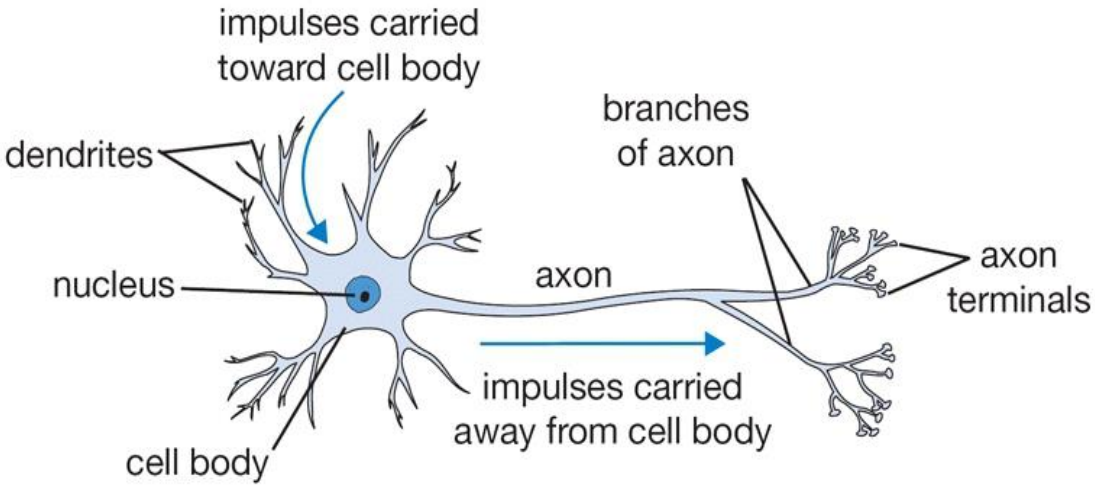



**Figure 24 Basic features of biological neurons [69]**

Most neurons comprise of three fundamental parts: i) dendrites which act as receptive zones and collect inputs from other neurons, or from external stimulus, ii) a soma (cell body) which imparts a nonlinear process, and finally ii) an axon, a cable like wire along which the output signal is transmitted to other neurons further down the processing chain. The connection site between two neurons is called a synapse. Synapses are elementary structural and functional units that mediate the interconnections between neurons. The signal source of most real neurons is chemical and impart short bursts of electrical activity. In Artificial Neural Networks, this electrical activity is modelled by a continuous variable $X_j$ which can be considered as a temporally average pulse. The majority of neurons encode their outputs as a series of brief voltage bursts. A biological neuron can have as many as 10,000 different inputs, and may send its output to many other neurons (up to 200,000). The same mechanism and function exist in ANNs. They have many very simple processors, each possibly having a local memory, which are organized in layers and are connected by weighted links. To achieve good performance, ANN employ a mass interconnection of simple computational cells, referred to as 'processing units'.

### 2.5.3 Components of Neural Networks

As previously described, the transmission of a signal from one neuron to another through synapses releases specific transmitter substances. The outcome is to lower or raise the electrical potential inside the receiving cell. Once the electrical potential reaches some threshold, the neuron will fire. This is the same principle that an Artificial Neuron is based on. In this way neural networks can realise a random mapping of one vector space onto another vector space [70].

A typical structure of an Artificial Neuron is shown in Figure 25 [71] [72].

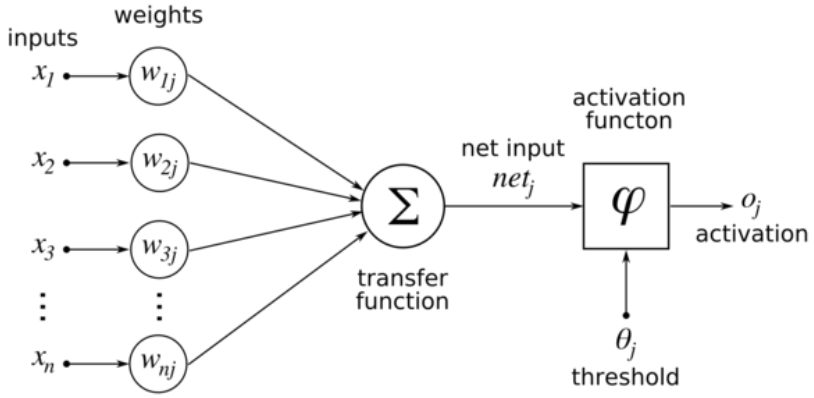



**Figure 25 Artificial Neuron Structure**

As shown, the neuron has *n* inputs symbolized with $i_1, i_2, i_j \dots i_n$ , which are connected to the neuron via multiplicative weights, $w_{1j}, w_{2j}, w_{3j} \dots w_{nj}$. The activation function represents a graded potential given by:

$$\varphi = \sum_{j=0}^{n} w_j u_j + \theta \qquad (12)$$

where, $\theta$ is the threshold level of the artificial neuron.

To realise functions that a single artificial neuron cannot perform alone, a chain of artificial neurons is created where the output of some neurons are connected to the input of others - forming a neural network. When designing such a network, equation (16) is modified by expressing the activation function of the $n^{th}$ neuron as:

$$\varphi_i = \sum_{j=0}^{N} w_{ji} x_j + \theta_i \qquad (13)$$

where, $x_j$ is either the output of another neuron or an external input. Neural Networks are formed in layers, with the first layer termed the input layer, the last layer is termed the output layer and all layers between them are called hidden layers. When there is only an input layer and a single layer of neurons constituting the output layer, it is called a single layer network. If there exist one or more hidden layers, they are called multilayer networks [71] [72]. The output of a neuron is set +1 if the weighted sum is >= 0, and -1 if the weighted sum is < 0.

An advantage of employing ANNs is that they can be considered as a "black box", and it is not necessary to have detailed a-priori information about the system. It also has the ability to manage large amounts of data using 'binning' to remove information of minor importance to the overall network model. The major advantage of neural networks is that they are able to make accurate models using unknown information hidden in the data. This process is termed 'network training'. There are two main training processes i) supervised and ii) unsupervised training. Training is considered as Supervised when the neural network has knowledge of the desired output and adjusts the weights so that the outputs are as close as possible to what is desired. Training is considered Unsupervised learning when the actual output is unknown and the system is provided only with a set of generic facts [73] [74].

### 2.5.4 Different structures of Neural Networks

There are various ways to categories neural networks, one of them being based on the technique used during the learning process. Furthermore, the interconnection architecture can be very different, and this provides another method to categorize them [68]. Different types of neural networks each have their relative merits for a particular application. It is widely acknowledged that there is no single method, statistical or neural network, which gives the best result for all types of problems. Generally, neural network architectures are classified as being either feedforward or recurrent. However, the underlying principles are similar. Each neuron in the network receives input signals and is connected to at least one neuron, with the connection being evaluated by a real number or 'weight'. The weights reflect the degree of importance of the given connection. In a feedforward network, signals are unidirectional, from input to output, whereas in a recurrent network each signal can be bidirectional by introducing feedback loops or cycles into the network structure.

### 2.5.5 Architecture of neural networks

- **Feedforward networks**

Neural networks are often arranged in layers such that the connections only exist between consecutive layers, all in the same direction. Such architectures are termed feedforward neural networks. They can possess any number of layers, units per layer, network inputs, and network outputs. Input signals propagate through the network in a forward direction, on a layer-by-layer basis, hence the term feedforward. The output is only a function of the current input, not of the past or future inputs or outputs. Node equations are therefore memoryless.

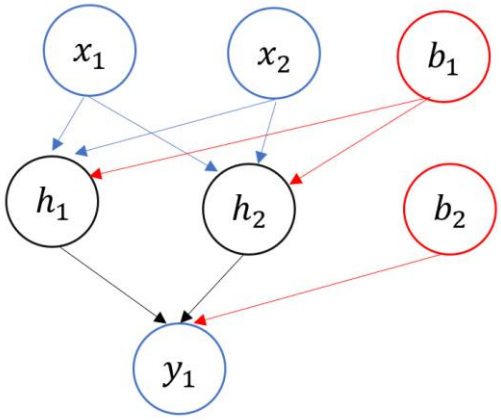



**Figure 26 Feed-forward Neural Network**

As can be seen from Figure 26, the neural network is built from input layers ($x_1$ and $x_2$), hidden layers ($h_1$and $h_2$), bias layers ($b_1$ and $b_2$) and a single output layer ($y_1$). Each of the layers also carries a weight ($w$). This weight changes as the neural network is trained. The output is calculated using:

$$y_1 = a((x_1 \cdot w_1) + (x_2 \cdot w_2) + w_3 \quad (14)$$

- **Recurrent Neural Network**

A recurrent Neural Network (RNN) is considered as an enhanced ANN architecture, and thus a variant of Elman's and Jordan's network. The ability to form more complex calculations than the static feed forward network is the reason behind adopting the RNN algorithm. Furthermore, the capability of learning temporal pattern sequences which are context or time-dependent is also an advantage of utilising the method. Embodying a short-term memory by activating a feedback network is also one of the main features of a simple Recurrent Neural Network (RNN). According to Tenti [75], requiring more substantial memory and connections in simulations, in comparison with a backpropagation network, is one of the main disadvantages of RNN, and

leads to relatively high computational overhead. However, utilising a RNN can yield better results.

The RNN can be fully or partially connected. In a fully connected RNN all the units are connected recurrently, whereas in partially RNN the recurrent connections are only partial. Fully connected RNNs use unconstrained fully interconnected architectures and learning algorithms that can deal with time-varying input and/or output in non-trivial ways (Omlin and Giles [76]). Full RNNs, have both feedforward and feedback connections, all of which are trainable. The network can take on any arbitrary topology as any node in the network may be linked with any other node, including the node itself. The only requirement to be made is that the network should have clearly defined input and output nodes. In [76] partial and full RNNs are used to classify strings with arbitrary length. Meanwhile, research done by Moody et al. [77] used a full RNN model to perform a spatial delayed matching to sample a task.

Partial RNNs are a type of feedforward network with the incorporation of a 'context unit' which stores the output from the hidden or output layers. Connections in partial RNNs are mainly feedforward but include a carefully chosen set of feedback connections. This network has all its feedforward connections trainable, whereas the feedback connections are fixed. Recurrence in partial RNNs allows the network to store cues from the recent past but does not appreciably complicate the structure and training of the whole network. Several models of partial RNNs exist, including the Jordan network (first introduced by Jordan in 1986) [78], and the Elman network (first introduced by Elman in his paper in 1990) [79]. Two types of local feedback are used in partial RNNs; the activation feedback (mainly applied in the Elman network), and output feedback (used in the Jordan network). Figures 27 (a) and (b) show the architecture of the Elman and the Jordan networks, respectively. The Elman network is a two-layer network with feedback from the hidden layer to the input layer, as depicted in Figure 27 (a). This recurrent connection allows the Elman network to both detect and generate time-varying

patterns. The input layer is divided into two parts; actual input units, and context units. Context units are connected to the forward direction with weights fixed to unity and are not trainable. The presence of this simple loop implies that the activation of hidden units at time $[t]$ can influence the activations of the hidden units at time $[t + n]$. Recurrent connections allow the network's hidden units to see its own previous output, so that the subsequent behaviour can be shaped by previous responses—effectively giving the network memory. In the case of the Jordan network, the architecture is realized by adding recurrent links from the network's output layer to a set of context units which form a context layer. Additionally, the context units are connected to each other and with themselves. This allows their next state to be calculated as a function of the current net output, their current state, and the current state of the other state units. The self-connections in the context layer give the context units some individual memory, or inertia. Hence, the Jordan network can be trained to recognize and distinguish between different input sequences [80].

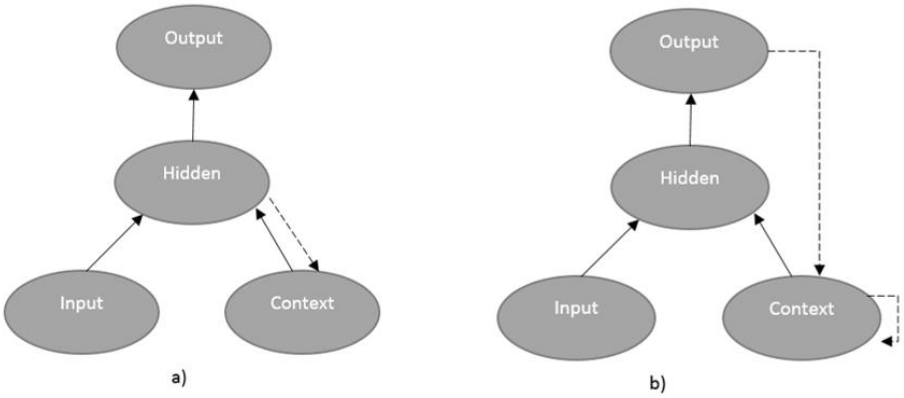



**Figure 27 Neural network structures a) Elman b) Jordan**

Another example of the recurrent neural network is the Hopfield variant [81]. Hopfield's approach provides a network structure where all the artificial neurons are fully connected. Like

every other neural network, HNN's have N inputs, each with an associated weight. A simplified structure of the Hopfield Neural network is shown in Figure 28.

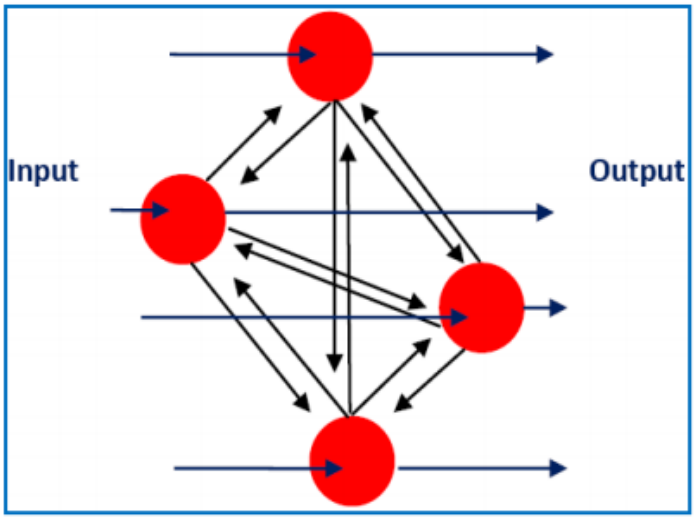



**Figure 28 Simple structure of Hopfield Network [82]**

Its architecture consists of a two-dimensional connected neural network in which the linking strengths between neurons, which are binary threshold neurons, are determined based on the constraints and solution criteria of the optimization problem to be solved. Each unit is connected to all other units except itself, and as a result, avoids a permanent feedback of its own state value [82].

This type neural network has two forms of learning rules in order to train the network's weights,

1) Asynchronously- where the weighted input sum is calculated and updated immediately, and

2) Synchronously- when the weighted input sums of all neurons are calculated without the update of neurons.

The original model used two-state threshold neurons where each neuron has two states characterized by the output of the neuron [81]. This output takes values of 0 or 1. Hopfield

derived Lyapunov functions for the network to check for stability and used it as a content-addressable memory. A discrete model of the neurons is as follow:

$$X_{ij}(k) = X_{ij}(k-1) + \Delta t \left[ \sum_{k=1}^{N} \sum_{l=1}^{N} W_{ijkl} Y_{kl}(k-1) + I_{ij} \right] \quad (15)$$

where $n$ is the number of nodes, $L$ is the set of connecting links in the network, $W$ is the weight of links, where $X_{ij}(k)$ is the internal state or local voltage of neuron, $Y_{kl}(t)$ is the output voltage of neuron $(i, j)$ in the network and $i \neq j$means there is no neuron on diagonal locations.

HNN's have found application for modelling, optimization, pattern recognition, signal processing, image processing and optical networks [83] [84] [85] [86].

Another recurrent neural network is based on Self-Organizing Maps (SOMs). Self-organizing networks or self-organizing maps were first reported by Teuvo Kohonen, in 1982 [87]. SOM training is usually considered unsupervised as there are no known target outputs associated with each input pattern—it processes the input patterns and clusters them through the adjustments of weights.

- **Neuron Models**

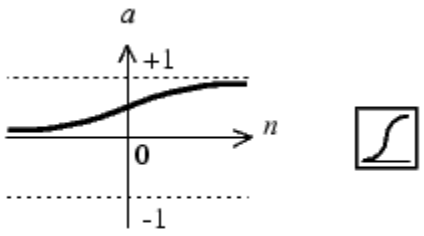
Neurons from adjacent layers have weighted connections between them and can employ one of a variety of transfer functions *f* to generate their outputs. In multilayer feed-forward networks, commonly used transfer functions are log sigmoid (*logsig*) transfer function, tan-sigmoid (*tansig*) transfer function and the linear transfer function (*purelin)* [58], as shown in Table II.

**Table II Mathematical Definitions of the activation function**

| Function | Definition | Range |
|---|---|---|
| Log-sigmoid | $f(x) = \frac{1}{1+e^{-x}}$ | $(0, +1)$ |
| Tan-sigmoid | $f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$ | $(-1, +1)$ |
| Linear | $f(x) = x$ | $(-\infty, +\infty)$ |

i. Log-sigmoid transfer function

Figure 29 shows that the output generated by the function *logsig* is between 0 and 1, while the neuron's input range can be ±∞.

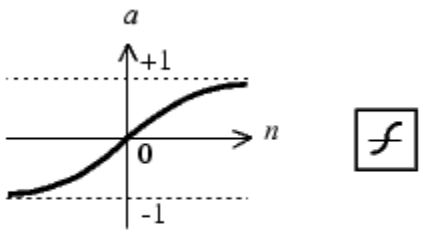



$\alpha = logsig(n)$

**Figure 29 Logsig function in neural networks**

ii. Tan-sigmoid transfer function

On the other hand, tan-sigmoid (or hyperbolic tangent) transfer function, which is shown in Figure 30, can be applied on the neurons to generate an output between -1 and +1 while the neuron's input range that can again be ±∞.

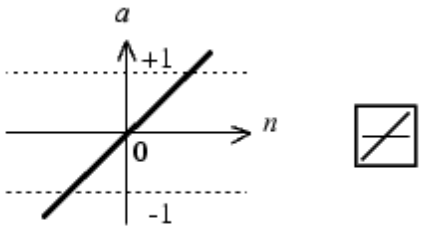



$$\alpha = tansig(n)$$

**Figure 30 Tansig function in neural networks**

iii. Linear Transfer function

As seen in Figure 29 and 30, the output value will be limited within a certain range when *logsig* or *tansig* transfer functions are applied on the last layer of networks. However, when using a linear transfer function *purelin,* shown in Figure 31, the network outputs can take on any value.




$$\alpha = purelin(n)$$

**Figure 31 Purelin function in neural networks**

The three described transfer functions are the most commonly used for back-propagation networks. However, there are other alternatives that can be used.

- Back-propagation training algorithm

To facilitate the production of meaningful forecasts, a neural network has to be trained using a procedure to perform the learning process. Back-Propagation (BP) has emerged as the most popular algorithm for supervised learning.

The underlying principle is to calculate the influence of each weight in the network using an iterative process based on gradient descent, where weights are adjusted in the steepest descent direction (negative of the gradient). Training of the network is performed in such a way that the weights are adjusted after the presentation of each single, or batch of training example(s). During the iterative process, two sets of signals are passed through the networks:

- **Function signals**: the input examples propagate through the hidden units and are processed by their activation functions, emerging as an output.
- **Error signals**: the errors at the output nodes are propagated backward layer-by layer through the networks, so that each node returns its error back to the nodes in the previous layer

The weights are determined using backpropagation by building connections among the nodes based on training data, producing a least-mean-square error measure of the actual or desired values from the output of the neural network. Initial values are assigned for the connection weights. To update the weights, the error between the predicted and actual output values is back propagated via the network [88]. Figure 32 illustrates the architecture of a backpropagation network, where $x_j$ (j = 1, 2... n) represent the input variables; $z_i$ ($i$ = 1, 2... $m$) represent the outputs of neurons in the hidden layer; and $y_t$ ($t$ = 1, 2... $n$) represent the outputs of the neural network [89]. In theory the neural network has the ability to simulate any kind of data pattern given sufficient training.

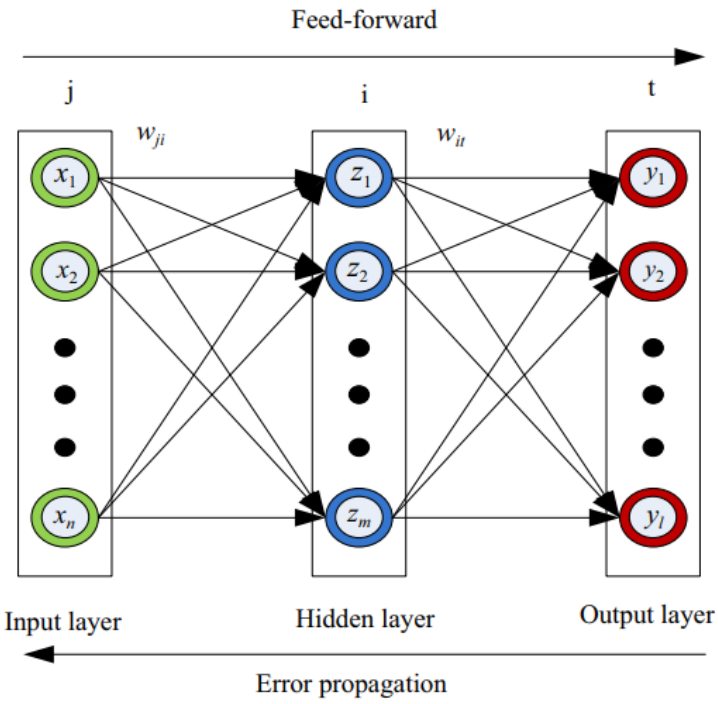



**Figure 32 A three-layer feed-forward back propagation neural network [89]**

## 2.6 Other regression techniques

### 2.6.1 ANFIS

The Adaptive Neuro-Fuzzy Interface System (ANFIS) was first proposed by Jang [90] in the 1990s. It is a multilayer feed-forward network that combines the qualities of a fuzzy-inference system and artificial neural network (ANN) to develop models with optimized rules and membership functions.

The Takagi-Sugeno (T-S) fuzzy system is a powerful tool for modelling and control in complex systems and has proven to be an efficient solution for non-linear system modelling [91]. An ANFIS is a hybrid algorithm that combines the high-level reasoning capability of Fuzzy Inference Systems (FIS), and the high-level power of pattern recognition of Artificial Neural (AN) Networks.

The ANFIS architecture consists of 5 layers with each layer performing a specific action. An example architecture with two inputs and one output is shown in Figure 33.

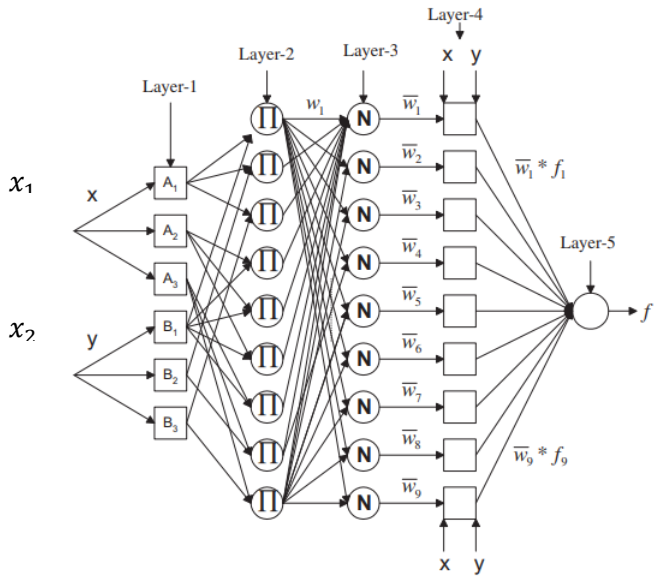



**Figure 33 ANFIS architecture of two inputs and nine rules [92]**

Layer 1 contains the data inputs. Layer 2 is the 'If' layer, that converts any inputs to fuzzy inputs and generates the membership function (MF) for each of the inputs. The MF used in this study is *gbellmf*. The output of each node in this layer is given by Equation (16),

$$O_{1,i} = \mu A_i(x) \quad for\ i = 1,2.. \qquad O_{1,i} = \mu_{b_{i-2}}(x) \quad for\ i = 3,4.. \qquad (16)$$

where, $O_{1,i}$ is the membership grade of the input nodes, $x_1$ and $x_2$. In layer 3, each output generated from the second layer is normalized and the nodes are fixed. Layer 4 contains fixed nodes that calculate the ratios of the firing strengths of the rules. Finally, the total of all neurons in layer 4 form the output in layer 5. MFs are tuned using a hybrid algorithm that combines gradient descent and least square methods. The hybrid learning algorithm is used to identify variables in the ANFIS architecture and is split into two parts (forward pass and backward pass). The forward pass applies the least-squares method to find the consequent parameters in Layer 4. During the backward pass, the error signals propagate backward and the premise variables are updated by gradient descent [90].

ANFIS's main advantages are that it has the ability to capture nonlinear structures of a process, along with possessing an adaption capability and rapid learning capacity.

### 2.6.2 Hammerstein- Wiener Model

A useful and general class of nonlinear dynamical models are so-called block-oriented models that consist of configurations of linear dynamic blocks and nonlinear memoryless blocks. The simplest examples in this class are cascaded systems with the nonlinear block either preceding (Hammerstein model) or following (Wiener model) the linear block. The Hammerstein model was first discussed in [93], while the Wiener model has its roots in Wiener's interest in nonlinear system using Volterra expansions [94]. Recently, Hammerstein-Wiener models have been used in biomedical applications such as estimation of muscle force from electromyography signals [95].

A model where a nonlinear block both precedes and follows a linear dynamic system is called a Hammerstein-Wiener model and is illustrated in Figure 34 [96].

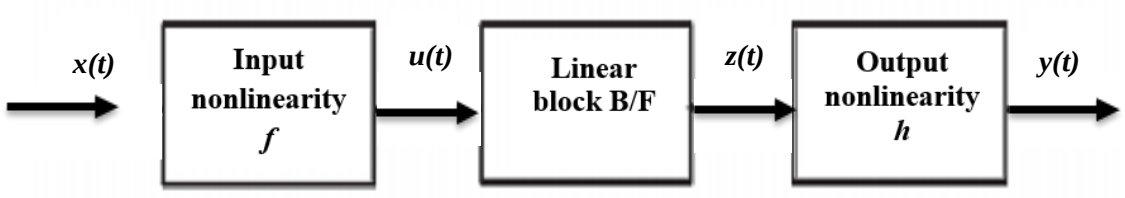



**Figure 34 The general Hammerstein-Wiener model structure, which consists of sandwiching a linear time invariant system between memoryless nonlinearities $f_f$ and $f_h$** [96]

Equation (17) is a nonlinear function that maps input data $u_t$:

$$u_t = f(x_t) \qquad (17)$$

And for the second block:

$$z_t = \frac{B}{F}\,(u_t) \qquad (18)$$

For outputs and inputs, the linear block is a transfer function matrix containing entries:

$$\frac{B_{ji}(q)}{F_{ji}(q)} \qquad (19)$$

where:

$j = 1,2,3 \ldots n_y$

$i = 1,2,3 \ldots n_u$

$q$ is the time shift operator

Finally, the third block:

$$y_t = h(z_t) \qquad (20)$$

is a nonlinear function that maps the output of the linear block to the system output. Since *f* acts on the input port of the linear block, this function is called the input nonlinearity. Similarly, since *h* acts on the output port of the linear block, this is called the output nonlinearity. If the system contains several inputs and outputs, functions *f* and *h* for each input and output signal are defined.

When a model contains only the input nonlinearity *f*, it is called a Hammerstein model. Similarly, when the model contains only the output nonlinearity *h*, it is called a Wiener model.

The Hammerstein-Wiener method calculates the output *y* in three stages [96]:

1. Calculate $u(t) = f(x(t))$ from the input data. The input nonlinearity is a static (memoryless) function, where the value of the output at given time $t$ depends only on the input value at time $t$. The input nonlinearity can be set as a sigmoid network, wavelet network, saturation, dead zone, piecewise linear function, one dimensional polynomial, or a custom network. It is possible to remove the input nonlinearity.

2. Calculate the output of the linear block using $u(t)$ and initial conditions. Configuration of the linear block is accomplished by specifying the numerator $B$ and denominator $F$ orders.

3. Calculate the model output by transforming the output of the linear block using the nonlinear function $h$.

The main benefit of these structures is to introduce fewer parameters to be estimated. Moreover, the polynomial representation has the advantage of providing additional flexibility and being easier to use.

### 2.6.3 Kalman filters

The Kalman filter, proposed by Kalman in 1960 [97] allows a unified approach for prediction of all processes that can be given a state space representation. A state space model is generally represented in two equations: Equation (21) is called the state equation which determines the state $X_{t+1}$ at time $(t + 1)$ using the previous state $X_t$ and a noise term, as shown below:

$$X_{t+1} = F_t X_t + W_t \quad , \quad t = 1, 2 \ldots \qquad (21)$$

where $F_t$ is a sequence of $v \times v$ matrices and $W_t$ is the process disturbance.

The Equation (22) is the observation equation which expresses the $w$ dimensional observation $Z_t$ as a function of a $v$ dimensional state variable $X_t$, and noise. Thus:

$$Z_t = G_t X_t + V_t, t = 1, 2 \ldots \quad (22)$$

where $V_t$ models measurement noise, $G_t$ is a set of $w \times v$ matrices and $W_t$ and $V_t$ are assumed uncorrelated.

If the underlying system is known, Kalman filters have successfully been used in a wide variety of application fields, for example, for predicting traffic flows [98] [99].

## 2.7 Summary

Of the methods reviewed it is important to select candidates specifically to address the problem of predicting neutrophil counts. The proceeding chapters now identify key characteristics that will be required of a candidate technique.

Due to the lack of information relating to the underlying system physics in neutrophil count prediction, and the highly non-linear characteristics of the system, Artificial Neural Networks are likely to provide an appropriate candidate for prediction. However, since this is a new prediction field, an initial feasibility study of other techniques is carried out for this application.

# Chapter III – Blood data pre-processing

## 3.1 Introduction

The blood-count dataset consists of inconsistence measures of constituent parts, and therefore requires some pre-processing to provide a consistent range of data values for the algorithms and importantly identify the most relevant blood constituents that are responsible for affecting neutrophil counts. The three methods used to pre-process the data are:

1. Normalising (scaling)- Different methods can handle only specific samples, for example, neural networks can only handle data that lie between 0 and 1. Therefore all data is scaled between 0 and 1.
2. Principal Component Analysis is used to remove unwanted characteristics in the data set (e.g. noise). It also provides a means of selecting the constituent blood variables that are most relevant to neutrophil counts.
3. The data is interpolated to provide more samples for training and testing.

## 3.2 Data interpolation

For this study, blood measurements are taken on a weekly basis and interpolation (using a cubic spline) is used to provide sufficient data to formulate a predictive model. The spline function consists of polynomial pieces on subintervals joined together using continuity conditions. The available measurement points are termed knots, and *f(x)* is a piecewise cubic function between consecutive knots $x_i$ given by

$$f(x) = \begin{cases} a_1x^3 + b_1x^2 + c_1x + d_1 & x\epsilon[x_0, x_1] \\ a_2x^3 + b_2x^2 + c_2x + d_2 & x\epsilon[x_0, x_1] \\ a_nx^3 + b_nx^2 + c_nx + d_n & x\epsilon[x_0, x_1] \end{cases} \quad (23)$$

For *f(x)* to provide a cubic spline, the following conditions need to be satisfied:

1) Function values are equal at interior knots.
2) The first and last functions pass through the end points.
3) First and second derivatives are equal at interior knots.
4) The second derivative is zero at the first point.

From $x$, $f(x)$ is solved for unknown values to provide a function that is smooth but without overfitting the data (unlike polynomial interpolation) [100].

## 3.3 Data normalization

Normalization is used to enhance the performances of certain processes that are sensitive to differences to the range of certain variables. For example, PCA (below) aims to capture the greatest proportion of variances, and as a result will give more weight to variables that exhibit the largest variances if feature normalization is not initially performed. Moreover, algorithms such as neural network can only work with variables that are in range between 0 and 1. Consequently, normalization is done to have the same range of values for each of the inputs to the artificial neural network model. This can promote the stable convergence of weights and biases.

Therefore, the measurement set of each variable in Table I is normalized in the range [0, 1] according to,

$$\hat{x} = \hat{x}_{min} + \frac{x - x_{min}}{x_{max} - x_{min}}(\hat{x}_{max} - \hat{x}_{min}) \quad (24)$$

where, $\boldsymbol{x_{min}}$ and $\boldsymbol{x_{max}}$ are taken from the data extrema. $\boldsymbol{\hat{x}_{min}}$ and $\boldsymbol{\hat{x}_{max}}$ are $\mathbf{10^{-5}}$ and 0.9 respectively to avoid zeros in the data.

To go back to non-normalized data, following formula is used:

$$y = y_{min} + \frac{\hat{y} - \hat{y}_{min}}{\hat{y}_{max} - \hat{y}_{min}} (y_{max} - y_{min}) \quad (25)$$

where, $\boldsymbol{y_{min}}$ and $\boldsymbol{y_{max}}$ are taken from the data extrema. $\boldsymbol{\hat{y}_{min}}$ and $\boldsymbol{\hat{y}_{max}}$ are $\mathbf{10^{-5}}$ and 0.9 respectively to avoid zeros in the data. $\boldsymbol{\hat{y}}$ is the point we are changing back into non-normalized datum point.

The reason Equations (24) and (25) are used is to normalize the data between (0.0001 and 0.9). Normalization of data within a uniform range is essential for two reasons:

(i) To prevent larger numbers from overriding smaller ones

(ii) To prevent premature overload of hidden nodes, which impedes the learning process. This is especially important when actual inputs take large values.

This normalization between slightly offset values will prevent slow or no learning.

## 3.4 Principal Component Analysis (PCA)

### 3.4.1 Introduction

Principal component analysis (PCA) is central to the study of multivariate data. Although it is one of the earliest multivariate techniques, it continues to be the subject of much research, ranging from new model-based approaches to algorithmic ideas from neural networks. It is extremely versatile, with applications in many disciplines [101] such as in QSAR (Quantitative structure-activity relationship) studies [102], in an evaluation of molecular lipophilicity [103], exploration of molecular structure-property relationships [104], use in organic chemistry [105], in chromatography [106], in biodegradation relationships [107], and for clustering in amino acids [108].

### 3.4.2 PCA for the blood constituents

It is important to focus on blood constituents that have the most effect on neutrophil counts. However, using too few variables will reduce information about the time series, while too many variables will impart greater noise, leading to distortion of the time series.

PCA is therefore applied to identify the input variables that are related to Neutrophil counts one day ahead. A superficial selection of variables might exclude data of significance, thereby denying the resulting model of valuable information on the input-output mapping.

### 3.4.3 Underlying Principle

The main purpose of PCA is to remove excessive variables from further analysis. In short, PCA is a statistical procedure that uses orthogonal transformation to convert a number of observed variables into a smaller number of artificial variables, called principal components [101]. The resulting principal components are used for subsequent analyses. An orthogonal transformation is a linear transformation $T : V \to V$ which preserves a symmetric inner product. An orthogonal transformation preserves lengths of vectors and angles between vectors,

$$(v, w) = (Tv, Tw) \quad (26)$$

Orthogonal transformations may be represented using orthogonal matrices.

The set of orthonormal transformations forms the orthogonal group, and an orthonormal transformation can be realized by an orthogonal matrix. Any linear transformation in three dimensions [109]:

$$x_1' = a_{11}x_1 + a_{12}x_2 + a_{13}x_3 \quad (27)$$

$$x_2' = a_{21}x_1 + a_{22}x_2 + a_{23}x_3$$

$$x_3' = a_{31}x_1 + a_{32}x_2 + a_{33}x_3$$

satisfying the orthogonally condition

$$a_{ij}a_{ik} = \delta_{jk} \quad (28)$$

where $\delta$ is a Kroncker delta, is an orthogonal transformation.

For a given *p*-dimensional data set *X*, the *m* principal axes $T_1, T_2, \dots, T_m$, where $1 \leq m \leq p$, are orthonormal axes onto which the retained variances is a maximum in the projected space [110]. A matrix $T^T$ ('$T$' denoting the transpose operator) is constructed by noting the *m* leading eigenvectors of the sample covariance matrix:

$$S = \frac{1}{N}\sum_{i=1}^{N}(x_1 - \mu)^T(x_1 - \mu) \quad (29)$$

where $x_1 \in X, \mu$ is the sample mean and *N* is the number of samples, such that,

$$ST_i = V_iT_i, i \in 1, \dots, m \quad (30)$$

where $V_i$ is $i^{th}$ largest eigenvalue of *S*. The *m* principal components of a given observation vector $x \in X$ are then given by,

$$y = [y_1, y_2, \dots, y_m] = [T_1^T x, T_2^T, \dots, T_m^T] = T^T x \quad (31)$$

### 3.4.4 Description and results of the proposed method

The propose method uses PCA to perform feature selection. The same as feature extraction, feature selection has the same goal which is dimension reduction. If we say that *x* is an eigenvector of a covariance matrix of PCA, we know that the feature extraction result of an arbitrary sample *a* is [111]:

$$z = a^T S \quad (32)$$

Where $S = [x_1 \dots x_N]^T, a = [a_1 \dots a_N]^T$ and $N$ is the dimension of sample vectors.

The absolute value of $x_i$ is able to statical evaluate the contribution, to the feature extraction results, of the $i^{th}$ feature component of samples. The smaller the value of $x_i$ the less the contribution of the $i^{th}$ feature component samples. If this absolute number is small enough, removing it does not effect on the feature extraction results. If the feature components is small enough and has little importance for feature extraction, we can also speculate that in the original space this feature is also not important. Therefore, after calculating the covariance matrix using the original samples, eigenvalues and eigenvectors are found. The eigenvectors corresponding to the first largest eigenvalues are selected. The contribution is calculated to the feature extraction results using:

$$c_j = \sum_{p=1}^{m} |V_{pj}| \quad (33)$$

Where $V_{pj}$ denotes the $j^{th}$ entry of $V_p$ which is the eigenvalue. Calculations of equation (33) is sorted in a descending order and we store this order in $d_j$. If $c_v$ is the largest among them, then we donate $d_1 = v$, which means that the $v^{th}$ feature component of the original sample is the most important feature. Therefore last one will be the least important and this one can be deleted. Since this is not the usual approach, it has also been proven with using Pearson's correlation in 3.4.5.

Table IV shows the ranking of which variables have got the most importance on neutrophil counts one day ahead, with 1 being the most and 14 the least. Therefore, the most relevant data values to neutrophil counts are:

- White blood cells

- Neutrophils day before
- 6-MP
- Platelets

Those 4 are chosen, as they have the most correlation to the output data (neutrophil counts).

**Table III Contribution to the neutrophil count prediction of all blood constituents**

| | Haemoglobin | WBC | Platelets | RCC | Haematocrit | Neutrophils | MCH | MCHC | MCV | 6MP | Lymphocytes | Monocytes | Eosinophils | Basophils |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Neutrophil Counts + 1 | 9 | 2 | 4 | 10 | 7 | 1 | 13 | 6 | 14 | 3 | 11 | 8 | 5 | 12 |

### 3.4.5 Pearson correlation criteria

One of the simplest criteria that is used widely for feature selection is Pearson correlation criteria. This is a good way to select variables from the input which can efficiently describe the input data while reducing effects from noise or irrelevant variables. This method is measuring the relevance of each feature with the output label, in this case neutrophil counts. Pearson correlation criteria is defined by:

$$R_i = \frac{cov(x_i, Y)}{\sqrt{var(x_i) * var(Y)}} \quad (34)$$

Where $x_i \in X$ is the candidate feature and regression target $Y$. A good individual feature is highly correlated with the target so correlation measures can be used for ranking or subset selection. The results of this method can be found in Table IV.

**Table IV Pearson correlation**

| | Haemoglobin | WBC | Platelets | RCC | Haematocrit | Neutrophils | MCH | MCHC | MCV | 6MP | Lymphocytes | Monocytes | Eosinophils | Basophils | Neutrophil Counts +1 day |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Neutrophil Counts + 1 day | 0.063284 | 0.942447 | 0.417519 | 0.058588 | 0.099903 | 0.983445 | 0.005779 | 0.152944 | 0.000001 | 0.688116 | -0.03393 | 0.086346 | 0.212093 | -0.02933 | 1 |

Pearson's r can range from -1 to 1. An r of -1 indicates a perfect negative linear relationship between variables, an r of 0 indicates no linear relationship between variables, and an r of 1 indicates a perfect positive linear relationship between variables. Looking at Table IV, the Pearson's correlation confirms the results of PCA which showed that the most important inputs are white blood cells, neutrophils day before, 6-mercaptopurine and platelets.

Another simple method called the backward elimination was used to prove the most influential factors as inputs by removing different factors from the input set, then training and testing a simple neural network (in this case NARX). If removing a particular input does not affect the output prediction accuracy, the input is removed. Removing the most significant features resulted in the biggest decline in prediction accuracy. This also showed that the most influencer inputs are the once also chosen by PCA and Pearson correlation criteria. This is not a method that should be used as an only method as taking inputs out will change the architecture of the neural network too. However, it's a good way to re-assuring the inputs chosen are correct.

## 3.5 Summary

Normalization and interpolation of data is undertaken to ensure that all blood count constituents receive equal importance and to make sure there is sufficient data for the training and testing of

the algorithms. In this case, PCA is used to reduce a set of 15 original variables to four main components viz. white blood cells, neutrophils day before, 6-mercaptopurine and platelets. The choice of PCA was also confirm using Pearson correlation criteria and backward elimination.

# Chapter IV – Initial feasibility study to predict neutrophil counts using NARX

## 4.1 Introduction

Predicting neutrophil counts is a non-trivial task due to the intrinsic complexity of the human body and associated blood data. While many time series may be approximated with a high degree of confidence, blood data are considered among the most difficult to be analysed and predicted. Furthermore, no previous work on neutrophil prediction exists in the literature. The underpinning non-linearity relates to the fact that blood data are affected by many highly interrelated factors, and these factors interact with each other in a very complex manner.

The motivation of this chapter is to present initial results using a relatively simple and well-known neural network architecture to determine whether neutrophil count prediction in leukaemia is feasible.

Various methods and techniques for the prediction of time series have been developed based on statistics and artificial intelligence. However, most prediction methods are complex with no inherent learning. Neural networks are powerful forecasting tools that draw on the most recent developments in artificial intelligent research. They are nonlinear models that can be trained to map past and future values of time series data and thereby extract hidden structures and relationships that govern the data [112]. Using neural networks, complex relationships between input and output variables can be learned by machines without requiring a human to specify the nature of the relationship. Neural networks have appeared as a powerful learning technique to perform complex task in highly nonlinear dynamic environments of time series. The application of neural networks in time series prediction has shown better performance in comparison to statistical methods because of their inherent ability to accommodate nonlinearities. In addition, it has been shown that neural networks are universal approximates and have the ability to produce complex nonlinear mappings [113]. The neural network algorithm chosen for this

initial investigation is the Nonlinear Autoregressive Exogenous (NARX) as it is one of the simplest and more robust techniques.

## 4.2 NARX Model Description

This section constitutes the first application of prediction algorithms to neutrophil counts. A recurrent neural network architecture is used based on NARX models. A three-layer NARX model is proposed for the application of predicting short-term (day ahead prediction) neutrophil counts in ALL childhood cancer.

NARX Neural Networks provide a general, practical method for learning discrete-valued, real-valued, and vector valued functions from example data. In what follows, the output signal of the network is derived from the input vector of the network using delay operators. The mathematical formulation of the output is expressed as:

$$\begin{aligned} &y(n+1) = f[y(n), y(n-1), \ldots, \\ &y\big(n-d_y+1\big);\ x(n), x(n-1), \ldots, \\ &x(n-\ d_x+1)] = f[y(n); x(n); W] \end{aligned} \qquad (35)$$

where $x(n)$ and $y(n)$ are the components of the input and output vector, respectively, and $d_x$ and $d_y$ are delays, $W$ is the matrix of the adjustable weights and $f$ is the unknown nonlinear function.

## 4.3 NARX architecture topology and parameters

To design an effective topology the following parameters need to be selected: the number of layers (more exactly the number of hidden layers, as the input and output layer is given); number of neurons in each layer, and the connections between neurons from different layers.

NARX is a recurrent dynamic network, with feedback connections enclosing several layers of the network. Where the next value of the dependent output signal is regressed on previous values of the output signal and previous values of an independent (exogenous) input signal. NARX model can be implemented by using a feed-forward neural network to approximate the function, and using the delays if a multi-step ahead prediction is done. The output can be considered of the NARX network to be an estimate of the output of some nonlinear dynamic system is modelled. The output is fed back to the input of the feed-forward neural network as part of the standard NARX architecture- parallel architecture. However, as the true output is available during the training of the network, series-parallel architecture is created, in which the true output is used instead of feeding back the estimated output, as shown in the Fig.

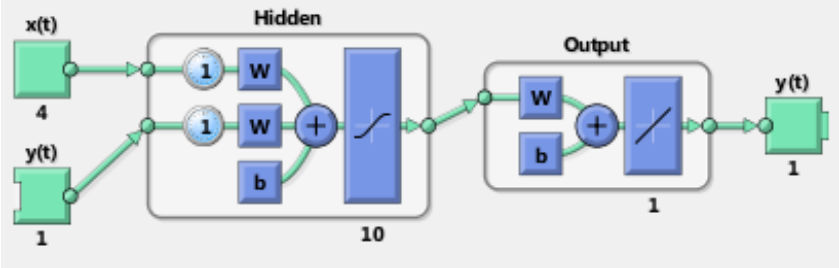



**Figure 35 NARX Series-parallel architecture**

Therefore, a series-parallel NARX architecture is used since the known output is available and it can be used instead of feeding back the estimated output. This form combines the features of a feed-forward network and dynamic networks. Static back-propagation is used to train the network to identify the underpinning nonlinear model.

The NARX model begins as a 'black box' with no information. Provisional tests were therefore undertaken to identify appropriate structural characteristics. It has been found that incorporating two hidden layers provide a good starting point for the proposed NARX neural network. The 4 inputs, as identified earlier using PCA, (WBC, platelets, 6MP and current neutrophil counts) are used as input to the series-parallel NARX network, whereas the next day neutrophil counts

are used as the actual outputs, so that the training process is more rapid and accurate. The network is fully connected, i.e. every unit is connected to each unit from the next layer. More connections (degrees of freedom) would allow the network to adapt to noise, but overlearning decreases the generalization capabilities.

The number of epochs used is 1000. To prevent overfitting, 6 validation checks were used, i.e. the learning stops if the error over the validation set increases. Periodically, while training on the learning data set the network is tested for performance on the cross validation set. If the network begins to over-train on the training data, the validation performance will begin to degrade. Thus, the validation data set is used to determine when the network has been trained as well as possible without overtraining.

## 4.4 Daily prediction results

To improve results it is traditionally recommended to use as much data as possible for training and testing. In the case of the daily observations, approximately four months of training data has been found to be the minimum threshold for training the NARX ANN.

In this case, 70% of the data is used for training and the remaining 30% is used for testing the performance of the resulting system. Half of the testing data is also used for cross-validation. Training datasets of this size have been used to produce all the results in the following graphs and tables. Specifically, three different training datasets were created from the normalized blood test data in the 4 months preceding days 113, 233 and 352. Figures 36, 37 and 38 graph the predictions of the subsequent 7 days one step ahead prediction. The dashed lines show the ANN's prediction of future values and the expected (known) values.

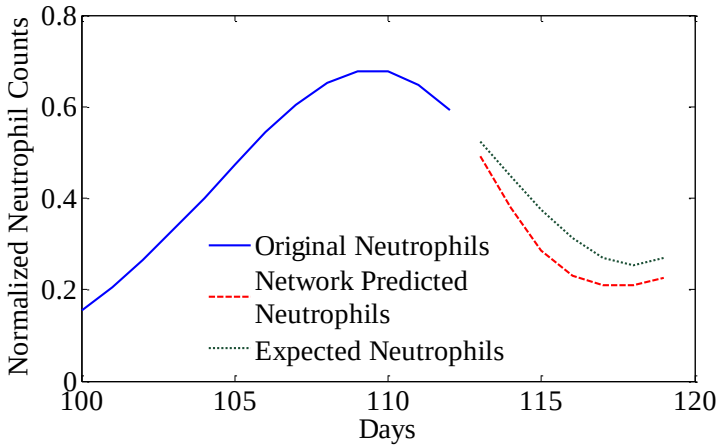



**Figure 36 Normalized neutrophil counts. Prediction period: 7 days**

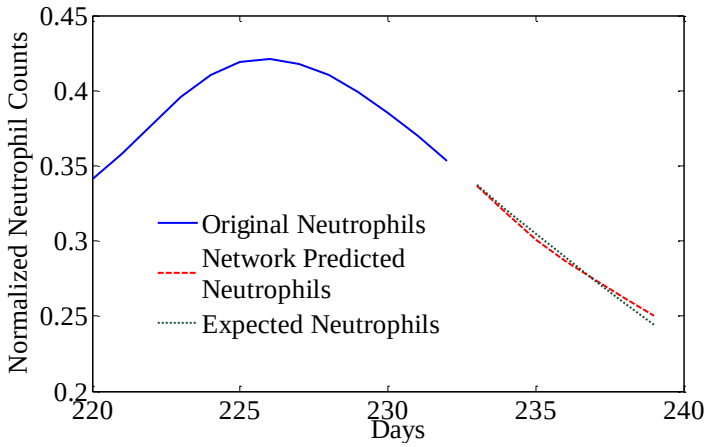



**Figure 37 Normalized neutrophil counts. Prediction period: 7 days**

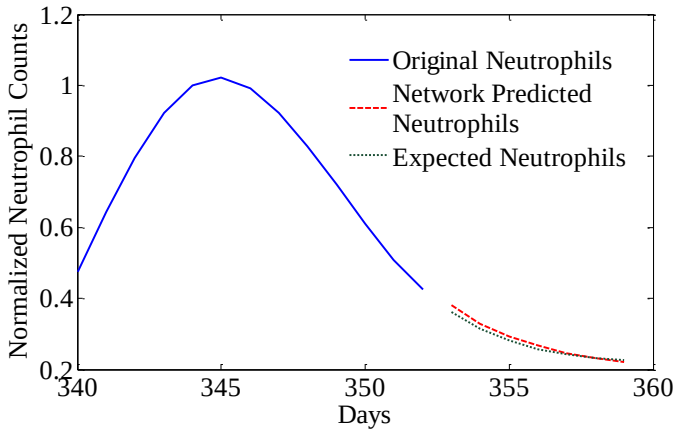



**Figure 38 Normalized neutrophil counts. Prediction period: 7 days**

The mean errors of the outputs from the NARX models are given in Table V for each of the three cases studies.

**Table V NARX error calculation**

| | Normalized error | | Error in Neutrophil prediction and 95% confidence intervals | | |
|---|---|---|---|---|---|
| ***Prediction algorithm*** | ***$10^3$ MSE*** | ***$10^2$ RMSE*** | ***Lower*** | ***Mean ($10^9$ / L)*** | ***Upper*** |
| NARX | 2.87 | 5.36 | -0.27 | 0.00523 | 0.285 |

## 4.5 Summary

This chapter has presented results of an initial investigation into the application of a NARX model to predict blood neutrophil counts. The results show effective one step ahead prediction performance using only modest training datasets (4 months historic data).

Predicting neutrophil counts with sufficient accuracy a week or more ahead is highly desirable and will be discussed in the next chapter. The potential is significant: aiding clinicians in reducing the risk of neutropenia thereby facilitating improved treatment success and reducing the number of Acute Lymphoblastic Leukaemia deaths brought about as a result of secondary infections.

# Chapter V – Model identification for multi-step ahead time series prediction of Neutrophil counts

## 5.1 Introduction

Here, the performance of various prediction methods are investigated and appraised. Whilst a large number of algorithms exist that have demonstrated high reliability for the future prediction of measurements, states or events, difficulties can arise when attempting to model systems that are highly non-linear, such as in drug prediction.

A simple test to show the non-linearity is used by getting a relationship between Neutrophils and WBC's, and Neutrophils and Platelets, using program "Eureqa". The relationship between two variables can only be described by using non-linear equations.

The formula that describes relationship between neutrophils and WBC is shown in (33), this equation however only describes 5% of the whole data, but it's the best fit found.

$$Neutrophils = WBC + 0.026 \cos\left(\frac{6.47WBC}{\cos(WBC) - 0.502}\right) \quad (36)$$

And formula that describes relationship between neutrophils and platelets is shown in (34). This formula describes 12% of data points.

$$\begin{aligned} Neutrophils \;=\; & 0.0701987087278242 \;+\; 0.310135918060874 \times Platelets \\ & \times Platelets^{cos(3.82618170062113 \times Platelets)} \times cos(2.02084747948388 \\ & \times Platelets \times cos(6.80393520184117 \times cos(4.10370940933091 \\ & \times Platelets))) \end{aligned} \quad (37)$$

Both (36) and (37) show that the data is not linearly correlated. Therefore, non-linear model estimation is needed.

Since, to the best of the author's knowledge, time series prediction has never been previously undertaken for neutrophil counts, a performance comparison of various techniques is now initially undertaken.

## 5.2 Comparison set-up

The data is initially divided into a training set and the model estimation set. However, unlike for the previous NARX case, the data set has now to be re-organized in a manner so that the prediction is given for $y_{t+7}$, i.e. provides a 1-week ahead prediction.

Algorithms are initially considered according to their application to neutrophil prediction problem based on the following criteria:

- They can cater for Multi-input, single-output systems (system has multiple inputs, i.e. blood count data, but a single output, i.e. neutrophil count).
- Prediction without *a priori* system models (no information on underlying system physics or physiology is assumed).
- Accuracy without the need for large amounts of training data (a practical system must provide high prediction accuracy without the need for a significant training period).
- Speed of training
- $p$ value when $h = o$ (The T-test is used to determine the significance of differences in prediction accuracy. The null hypothesis is that the algorithm in question is not significantly different in performance to other algorithms. If the $p$ -value is lower than the significance level (5%) as chosen, then the null hypotheses is rejected in favour of the alternative hypothesis. If the $p$ -value is greater than or equal to the significance level, then the null hypothesis is failed to be rejected. Thus, if $h$ is 1, the test rejects the null hypothesis at a 5% significance level in this case, and when $h$ is 0 the null hypothesis is confirmed.

This technique only shows whether there is difference among the model's performance or not).

## 5.3 Results of comparison of different prediction algorithms

For this study, many algorithms are identified and trialled in order to select those most suitable e.g. Artificial Neural Networks (ANNs), Support Vector Machine (SVM), Relevance Vector Machine (RVM), *k*-Nearest Neighbour (kNN), Autoregressive Integrated Moving Average (ARIMA), Analysis of Variances (ANOVA), regression trees, clustering techniques and bilinear models. Each algorithm is trained using the same programming language and environment – coded and analysed using MATLAB m-files and the MATLAB statistical toolbox. The ability for the algorithm to deal with Multi-input-single-output (MISO) systems, with unknown system dynamics are considered the most critical criterion, and those that do not address these are not considered further. The outcomes can be seen in Table VI.

**Table VI Algorithms comparison**

| | **Objectives** | | | | |
|---|---|---|---|---|---|
| **Methods** | **MISO system*** | **Without known system*** | **Prediction error** | **Speed** | ***p* value when *h*=o** |
| NARX Neural Network | ✓ | ✓ | **0.50** | **1 minute** | **80%** |
| Feed forward propagation Neural Network | ✓ | ✓ | **0.62** | **1 minute** | **66%** |

| | | | | | |
|---|---|---|---|---|---|
| MLP Neural Network (code) | ✓ | ✓ | **0.82** | **1 minute** | **75%** |
| MLP Neural Network (toolbox) | ✓ | ✓ | **0.38** | **1 minute** | **97%** |
| Elman Neural Network | ✓ | ✓ | **1.00** | **1 minute** | **43%** |
| Cascade-forward backrop NN | ✓ | ✓ | **1.56** | **1 minute** | **38%** |
| Competitive Kohonen NN | ✘ | | **N/A** | **N/A** | **N/A** |
| Hopfield NN | ✘ | | **N/A** | **N/A** | **N/A** |
| Layer recurrent NN | ✓ | ✓ | **0.81** | **1 minute** | **74%** |
| Self-organizing maps | ✘ | ✘ | **N/A** | **N/A** | **N/A** |
| SVM/SVR | ✓ | ✓ | **0.40** | **3 minutes** | **86%** |
| RVM/RVR | ✓ | ✓ | **0.69** | **2 minutes** | **51%** |
| LSSVM | ✓ | ✓ | **0.64** | **1 minute** | **60%** |
| K-NN | ✓ | ✓ | **1.39** | **1 minute** | **h rejected** |
| Gaussian Mixture Regression | | ✘ | **N/A** | **N/A** | **N/A** |

| | | | | | |
|---|---|---|---|---|---|
| Kalman Filter | | ✘ | **N/A** | **N/A** | **N/A** |
| ARIMA | ✘ | | **N/A** | **N/A** | **N/A** |
| LWPR | ✓ | ✓ | **0.80** | **1 minute** | **27%** |
| Gradient Boosting Decision Tree | ✓ | ✓ | **0.38** | **1 minute** | **25%** |
| Regression Tree | ✓ | ✓ | **0.52** | **1 minute** | **62%** |
| Multiple linear regression | ✓ | ✓ | **0.66** | **1 minute** | **64%** |
| Hidden Markov Model | ✘ | | **N/A** | **N/A** | **N/A** |
| ANFIS | ✓ | ✓ | **0.75** | **4 minutes** | **9%** |
| Hammerstein-Wiener Model | ✓ | ✓ | **0.88** | **1 minute** | **20%** |
| Bayesian network | ✘ | | **N/A** | **N/A** | **N/A** |

Results of the T-test are shown in Table VI. It can be seen that MLP and SVM have the lowest prediction error and the highest *p* value and hence these are taken forward for further development. The structure of both, MLP and SVM is defined in the next two chapters.

## Prediction performance comparison using error metrics

Having identified the favourable properties of MLP and SVM, their performance is compared by validating the models on two alternative data sets. These data is obtained from the same data

set but they are not seen by the algorithms before. First set is obtained at the beginning of the data and second set later on in the data. Graphic methods (which are figures in the three chapters) enable visual comparison of model responses and measured values as well as giving an initial insight of overall model quality. Numerical methods quantify the deviation of the model from the actual, desirable values, by means of statistical quality measures that can be basically divided into absolute and relative.

When evaluating the performance of the models, 14 measures were considered as metrics:

- Four absolute measurements: mean squared error (MSE), root mean square error (RMSE), sum squared error (SSE), the average absolute error (mean absolute error, MAE);
- Three relative measurements: Pearson coefficient ($r$) and coefficient of determinance ($r^2$) and index of agreement (D) and potential error (PE) that is used to calculate index of agreement. PE represents the biggest possible error that can occur for one period of model prediction and real values of modelled variables.

### 5.3.1 Absolute model quality metrics

Absolute quality measures add the differences between model responses and actual values with the result expressed in units of the modelled variables, and thus accurately express the deviation of the model from the ideal, and possible estimates of average deviation or total deviation. In both cases adding individual error values with positive and negative polarity results in their cancelation and can thereby give wrong information on model quality. Therefore, absolute or square values of individual errors are used, and then aggregated (or averaged). Most commonly used absolute measures are the mean square error (MSE), the root of the square quadratic error (RMSE), sum of square error (SSE) and mean absolute error (MAE).

The sum of the square error (**SSE**), defined in (38), measures the total deviation responds to the model in relation to the measured data, counting the first individual error for each sample ($i$) as the difference between actual, measured, desired value ($\hat{y}_i$) and response model ($y_i$), and then summing the squares of individual errors or the whole set of available data ($n$).

$$SSE = \sum_{i=1}^{n} (\hat{y}_i - y_i)^2 \quad (38)$$

The Mean Square Error (MSE), defined by the expression (39) is also based on square error, and account an average error for a set of N members available.

**MSE** is given with the following formula:

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2 \quad (39)$$

This is one of the most widely used error measures. The main advantages of MSE are:

- Its fast and easy to compute
- Its continuous
- It is very intuitive in that its simply an average error

As with all error metrics, MSE has its drawbacks. The first problem is that only the difference between the actual and the predicted values enters into the computation of MSE. The direction of the difference is ignored. For instance, consider a model being developed to predict the neutrophil counts a month from now. The clinician is intending to use this prediction to determine whether to increase or decrease 6-MP intake. If the model predicts a significant move in one direction, but the neutrophil counts move in the opposite direction, this is a significant error. This error correctly registers with MSE. However, other errors contributing to MSE are possible, and these errors may be of little or no consequence to this kind of prediction and would be more useful to consider in predictions such as price movements etc. Another drawback of

MSE is the squaring operation, which emphasizes large errors at the expense of those that are smaller.

SSE and MSE are considered quality measures as they express a deviation of the square of the modelled variables. For model estimation the most commonly used metric is the root mean square error (**RMSE**), defined in (40) and mean absolute error (**MAE**), (41).

$$RMSE = \frac{1}{n}\sum_{i=1}^{n}\sqrt{(\widehat{y_i} - y_i)^2} \quad (40)$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\widehat{y_i} - y_i| \quad (41)$$

In general, the lack of quality measures used by squaring an individual error, as in the case of SSE, MSE and RMSE, makes higher individual errors have a greater significance in the results. By contrast, the MAE calculates the mean value of the absolute error according to (41) and thus the same importance is given to errors of all values.

Using absolute quality metrics can measure the performance of the models and get accurate information on how much the model output deviates from the actual value of the variable that is being modelled, i.e. whose value is being predicted.

### 5.3.2 Relative model quality metrics

In order to be able to set general conditions of acceptable and unacceptable qualities of model performance, relative quality measures are also used. In particular, the Pearson coefficient ($r$) and the coefficient of determination ($r^2$) and the index of agreement ($D$) and potential error (PE). Output values of each of the methods, as defined in (42-45), range from a minimum 0 to

$\infty$ and represent the measure of correlation between the measured values ($\widehat{y_i}$) and the response of the model or the predicted values ($y_i$)—the exception is (40) which gives a percentage output where the higher the percentage the better the prediction.

**R-Squared** is inversely related to MSE. The advantage of R-squared over MSE lies purely in its interpretation. It is calculated by expressing the MSE as a fraction of the total variance of the dependent variable, then subtracting this fraction from 1.0, as shown in Equation (36).

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(\widehat{y_i} - y_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \quad (42)$$

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i \quad (43)$$

If a model is completely naive, always using the mean as its prediction, the numerator and denominator of (42) will be equal, and R-squared will be zero. If the model is perfect, always predicting the correct value of the dependent variable, the numerator will be zero, and R-squared will be one. This normalization to the range between zero and one makes interpreting R-squared easy, although pathologically poor predictions can result in this quantity being negative [114], and in reality, the values for R-squared can range from $-\infty$ for the worst and $+1$ for the best.

The values of the linear correlation coefficient ($r$) range from 0 to +1, with positive values implying the existence of positive correlation between the variables considered and their proportional change, or when the value of one variable decreases (or increases), the same happens with the other variable. Negative values of correlation coefficients imply the existence of negative correlation between the variables considered, which means that the change of the variables values is inversely related.

The index of agreement (*D*) is defined in (44), where PE is the potential error defined by the (45) and represents the largest possible error that can occur for a pair of predictions of the model and real values of modelled variables. Index of agreement (*D*) takes values to set [0, 1], with higher values pointing to a better matching of the model response to the actual values.

$$D = 1 - \frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{\sum_{i=1}^{n}(|y_i - \bar{y}| + |\hat{y}_i - \bar{y}|)^2} = 1 - N \cdot \frac{MSE}{PE} \quad (44)$$

$$PE = \sum_{i=1}^{n}(|y_i - \bar{y}| + |\hat{y}_i - \bar{y}|)^2 \quad (45)$$

Percentage error is one of the most common used methods for judging the quality of the model prediction and is given in (46). The range of percentage error varies from 0% for a very precise prediction to $+\infty$ for very poor predictions.

$$percent\ error = \frac{|(\hat{y}_i - y_i)|}{\hat{y}_i} x\ 100\% \quad (46)$$

## 5.4 Summary

This chapter has presented a review of candidate prediction algorithms that appear in the literature, comparing them according to the five metrics. All algorithms are trained with 5 months of data. Out of all the algorithms, only 2 classes were considered effective candidates based on the criteria viz. MLP neural network, and SVR. For completeness, Table VII (for SVR) and Table IX (for MLP) show both of the algorithms considered, and their relative merits.

# Chapter VI – Neutrophil count prediction using SVM

## 6.1Introduction to support vector machines

The Support Vector Machine (SVM) algorithm is a supervised learning technique which uses a hyperplane to separate classes of training data. The support vector method represents a classification model that is characterized by good generalization capabilities and the ability to successfully classify large dimensional data. Its theoretical basis is based on the statistical theory of learning (statistical learning theory) [47]. If there is no linear boundary, kernel functions are used (kernel functions) gain transformation into linearly separable space. The most commonly used kernel functions are: linear, polynomial and radial bases (radial base function, RBF), although any function that satisfies Mercer's condition can be used [115].

Due to the robustness of the model, SVM gains popularity in recent research and has been applied to environmental modeling, credit rating and consumer credit assessment, bank forecasting, bankruptcy forecasting, financial forecasts. Even though SVMs were developed to solve the classification problem, recently they have been extended to the domain of regression problems. The term SVR is typically used to describe regression with support vector methods.

## 6.2Calculation and learning of SVM algorithm

The training data consists of input objects and desired outputs. The Support Vector Regression (SVR) machine uses the same principles as SVM, but the output is a real number. SVR relies on a kernel function to map training data into a high-dimensional space. In this case, a radial basis (Gaussian) function is used instead of the typical linear dot product $x_1{}'x_2$ with Gaussian function as shown in (47):

$$G(x_1, x_2) = \exp(-\|x_1 - x_2\|^2) \quad (47)$$

The SVM model has been developed using a MATLAB toolbox. To train a model that accurately describes the dataset, experiments using different model hyper-parameters are initially undertaken. In the case of SVR, these parameters are the penalty parameter *C*, the Gaussian kernel parameter $\gamma$ and the epsilon value, which is chosen to be as small as 0.04.

Let $x(i,j)$ and $y(i)$ be the training predictor features and output response samples, respectively. In terms of SV regression, the goal is to find a function that has at most a deviation $\epsilon$ from the actual values $y(i)$. The problem can be configured as an optimization problem:

$$\min \frac{1}{2} w^T . w$$
$$s.t. \begin{cases} y_i - (w^T . \emptyset(x) + b) \leq \epsilon \\ (w^T . \emptyset(x) + b) - y_i \leq \epsilon \end{cases} \quad (48)$$

where $\emptyset(x)$ is the kernel function, *w* is the margin and $b$ is the bias value. Moreover, the principal of SVR is similar to SVM, in that, once trained the SVR will generate predictions using the following formula:

$$f(x) \equiv \sum_{i=1}^{l} \theta_i \emptyset(x, x_i) + b \quad (49)$$

## 6.3Results of SVM

The blood count data set is divided into two parts. The beginning of the data is used first with 4 months of training plus 7 days ahead 10 times, and then data in the middle with the same principle- 4 months of training data with 7 days ahead predictions 10 times.

### 6.3.1 SVR simulation and error results

The relative accuracy of a model must be checked on a set of data that was not used during the training phase. The model is evaluated based on its prediction errors. The error measures used for SVM are: % error, MSE, RMSE and R-squared (for their explanation refer to Chapter V).

The resulting SVM performance prediction is summarized in Figure 39, where it can be seen that maximum prediction errors occur in week 4 and the minimum prediction error in weeks 1, 2 and 5. The overall average accuracy of the algorithm is ~60%.

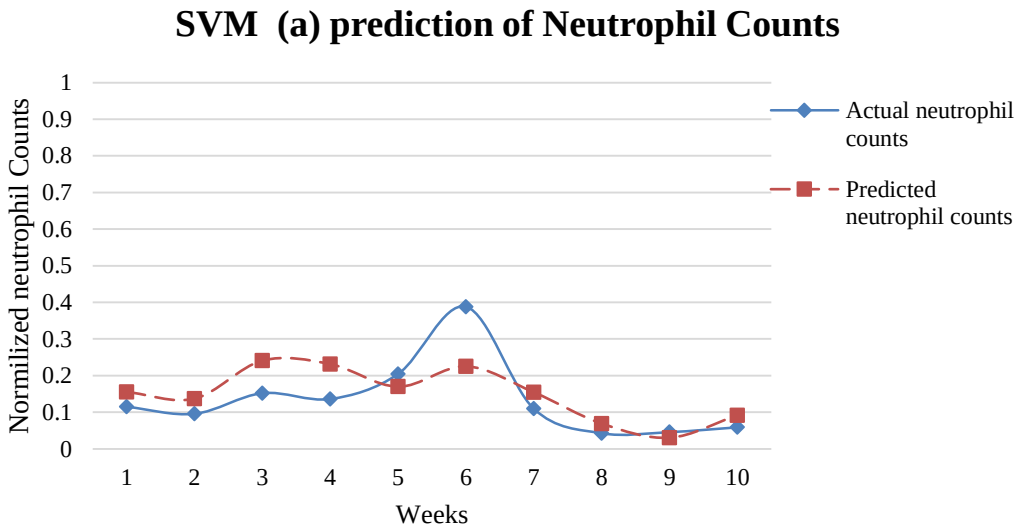



**Figure 39 Normalized neutrophil counts prediction (a) using SVM NN. Prediction period: 7 days ahead**

However, when training and measuring the performance on a later data set, Figure 40, it can be seen that the results are not as accurate.

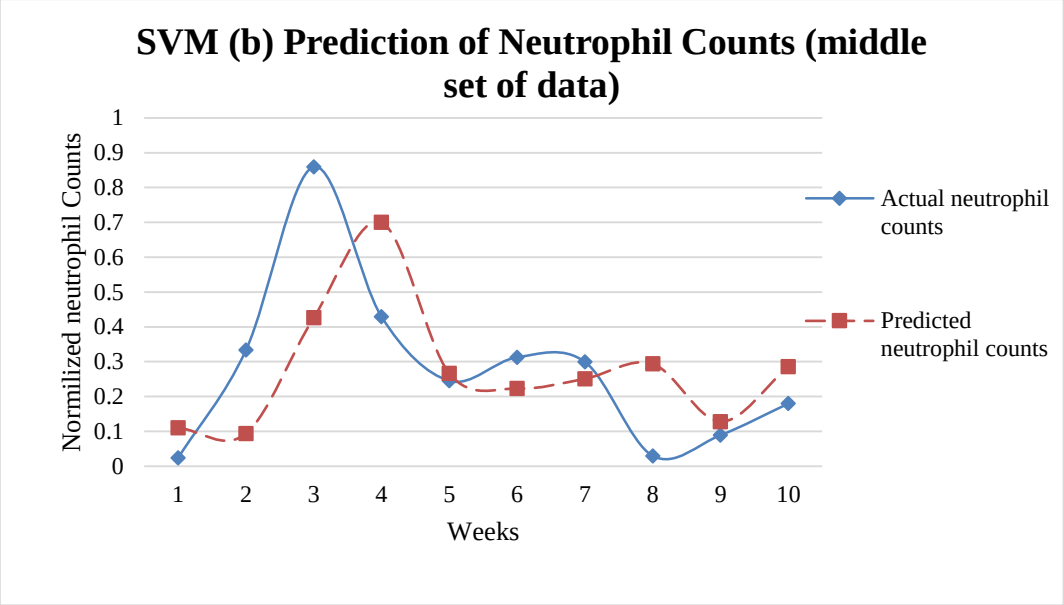



**Figure 40 Normalized neutrophil counts prediction (b) using SVM NN. Prediction period: 7 days ahead**

The errors are shown in Table VII and VIII, from which it can be concluded that SVM does not perform well with sufficient reliability.

**Table VII Error calculation for SVR models**

| Weeks | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **Mean** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **For (a)** | | | | | | | | | | | |
| **APE** | 35.28 | 43.24 | 58.44 | 70.28 | 16.92 | 41.91 | 41.11 | 62.19 | 33.41 | 56.50 | 45.93 |
| **SSE** | 0.0016 | 0.0017 | 0.0079 | 0.0091 | 0.0012 | 0.2645 | 0.0020 | 0.0007 | 0.0002 | 0.0011 | 0.0052 |
| **For (b)** | | | | | | | | | | | |
| **APE** | 364.9 | 71.87 | 50.38 | 63.13 | 8.567 | 28.74 | 16.17 | 888.5 | 43.43 | 58.90 | 159.5 |
| **SSE** | 0.0075 | 0.0574 | 0.1875 | 0.0734 | 0.0004 | 0.0081 | 0.0023 | 0.0696 | 0.0015 | 0.0112 | 0.4189 |

**Table VIII Error calculation for SVR models**

| | $RMSE$ | $MSE$ | $MAE$ | $R^2$ | $D$ |
|---|---|---|---|---|---|
| **SVM (a)** | 0.0583 | 0.0053 | 0.0583 | 0.4466 | 0.7794 |
| **SVM (b)** | 0.1597 | 0.0419 | 1.5976 | 0.2216 | 0.6779 |

It is clear that the first testing of SVM outperforms the second one. However, more generally looking at them both as a unit, it can be concluded that the % error is quite high. SSE is low in case (a) but high in (b), this might be because the data in general is higher in that set of data. The same counts for MSE and RMSE, which are the mean of SSE and the square root of that mean, respectively. The closer the $R^2$ is to $1$ or $-1$ the better the fit of the model. In this case, again, SVM seems a little unreliable. When looking at the value of D, here again the number nearer to 1 is better, and this value agrees with $R^2$. All the error measurements are showing consistent results.

### 6.3.2 Confidence Interval for SVM

When calculating confidence intervals, it is clearly beneficial to have the confidence interval as narrow as possible for any specified probability. In the cases presented here, the confidence interval is obtained using the following:

1. Find the mean of the differences between predicted and actual neutrophil counts
2. Calculate the standard deviation
3. Calculate the standard error by dividing the standard deviation by the square root of the sample size ($\sqrt{10}$)
4. Multiply the standard error by 1.96

5. Calculate the confidence interval by a) adding the margin of error to the mean (finding the upper limit) and b) subtracting the margin of error from the mean (finding the lower limit)

The 95% confidence interval for SVM is shown in Table IX and Figure 41.

**Table IX Confidence interval for SVR prediction**

| | **Error in Neutrophil prediction and 95% confidence intervals** | | |
|---|---|---|---|
| ***Prediction algorithm*** | ***Lower*** | ***Mean ($10^9$ / L)*** | ***Upper*** |
| SVR (a) | 0.03199 | 0.05833 | 0.08466 |
| SVR (b) | 0.08046 | 0.15976 | 0.23906 |

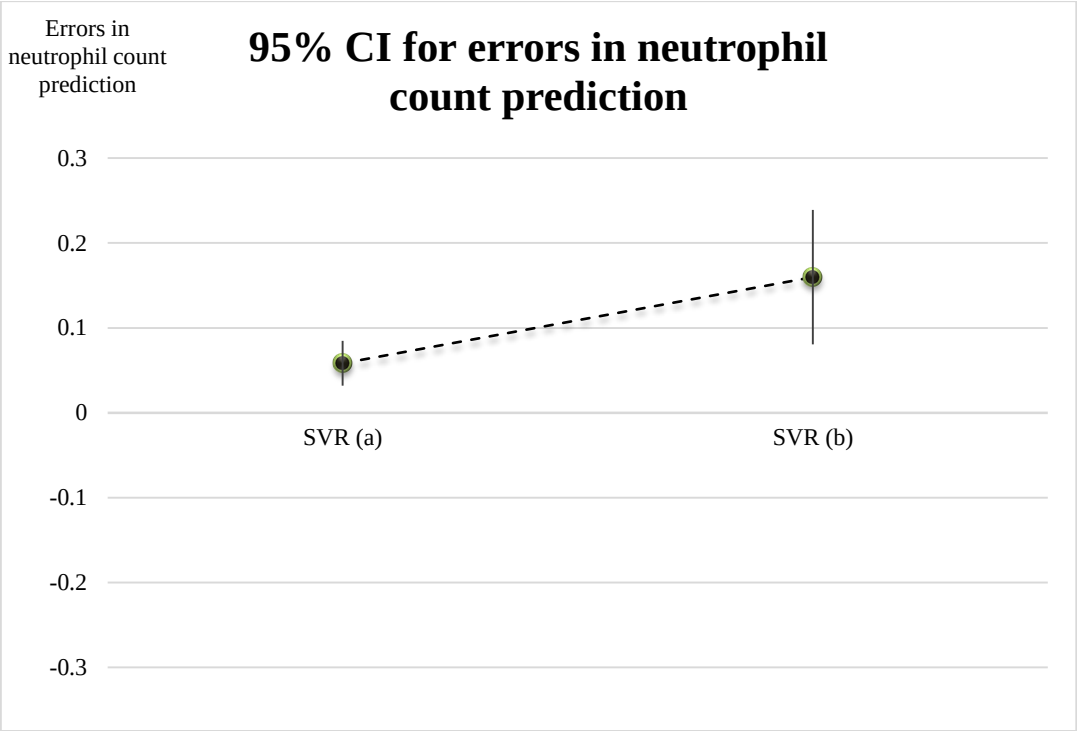



**Figure 41 Confidence interval for SVR prediction**

## 6.4Summary for SVR prediction

In this chapter, a novel approach to ALL drug dosing is proposed. SVR-based prediction models were chosen based on their relative merit from a suite of alternative techniques. The results show predictions of neutrophil counts over successive 7-day horizons using modest training datasets, and resulting in prediction accuracies of up to 60% in the first instance. However, in the second data set, the performance declines. Consequently, an alternative method, MLP is investigated in the next chapter in an attempt to improve prediction accuracy.

# Chapter VII – Neutrophil count prediction using MLP

## 7.1Introduction to Multi-layer perceptron

The MLP neural network has a feed-forward architecture that uses back-propagation for training, entailing a forward pass and a backward pass through all layers of the network. In the forward pass, the input is applied to the network, and its effect is circulated through each layer, ultimately producing an output. This output is subtracted from the actual output to produce an error. The effects of the error then propagate backwards through the network in order to reduce the error by adjusting the weights.

## 7.2Parameters Settings of MLP

As described in the previous chapters there is the choice of three activation functions for the neurons. Here, the sigmoidal neuron activation function is used with sloping factor $\boldsymbol{\alpha}$ in the output layer:

$$\varphi = \frac{1}{1 + e^{-\alpha j}} \qquad (50)$$

The coefficient α is related to the steepness of the sigmoidal function. With increasing α, the sigmoidal function defined by (50) approaches a hard limit. Initially, α is chosen to be 1, but as the training phase progresses, this changes as a result of updates from the error.

For the hidden layers, the *tanh* activation function was used with steepness parameter $\alpha$:

$$\varphi = \tanh(\alpha j) \qquad (51)$$

This combination of sigmoidal and *tanh* function was chosen as the optimum combination after trying many.

An optimal number of hidden neurons was considered to be the smallest number of neurons that allowed the network to correctly model the relationship between input and output data.

However, a generic method to determine the minimum required number of neurons has yet to be reported, and so trial and error, or rules of thumb are generally used. It is recommended to consider the lower $N_H$ sum of the inequalities in (52) and (53), where $N_I$ denotes the number of neuronal network inputs, and $N_S$ indicates the number of training samples.

$$N_H \leq 2N_i + 1 \quad (52)$$

$$N_H \leq \frac{N_s}{N_i + 1} \quad (53)$$

Equation (52) is used as the number of training samples increases with time. Since for this application there are 4 input, (52) gives us a maximum number of neurons in the first hidden layer as 9, and in the second layer as 19. After trial executions with different number of neurons in each hidden layer, the following architecture was chosen: one input layer (with 4 inputs), two hidden layers (with 6 neurons in first hidden layer and 3 neurons in second hidden layer), one output layer (with one output) and 3 biases, each connected to the input and hidden layers, as shown in Figure 42.

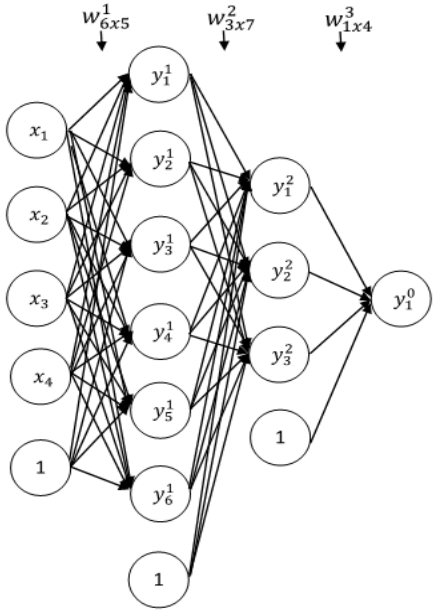



**Figure 42 MLP architecture**

In the feedforward pass, the weighted sums are calculated using [116],

$$k_1\begin{bmatrix} v_1^1 \\ \vdots \\ v_6^1 \end{bmatrix} = \left[\begin{pmatrix} w_{1,1}^1 & \cdots & w_{1,7}^1 \\ \vdots & \ddots & \vdots \\ w_{6,1}^1 & \cdots & w_{6,7}^1 \end{pmatrix}\right]\begin{bmatrix} x_1 \\ \vdots \\ x_6 \\ 1 \end{bmatrix} \xrightarrow{\varphi} \begin{bmatrix} y_1^1 \\ y_2^1 \\ \vdots \\ y_6^1 \end{bmatrix} = \begin{bmatrix} \varphi(v_1^1) \\ \vdots \\ \varphi(v_6^1) \end{bmatrix} \quad (54)$$

$$k_2\begin{bmatrix} v_1^2 \\ \vdots \\ v_3^2 \end{bmatrix} = \left[\begin{pmatrix} w_{1,1}^2 & \cdots & w_{1,7}^2 \\ \vdots & \ddots & \vdots \\ w_{3,1}^2 & \cdots & w_{3,7}^2 \end{pmatrix}\right]\begin{bmatrix} y_1^1 \\ \vdots \\ y_6^1 \\ 1 \end{bmatrix} \xrightarrow{\varphi} \begin{bmatrix} y_1^2 \\ y_2^2 \\ y_3^2 \end{bmatrix} = \begin{bmatrix} \varphi(v_1^1) \\ \vdots \\ \varphi(v_3^1) \end{bmatrix} \quad (55)$$

$$k_0[v_1^0] = \left[\begin{pmatrix} w_{1,1}^0 & \ldots & w_{1,4}^0 \end{pmatrix}\right]\begin{bmatrix} y_1^2 \\ \vdots \\ y_3^2 \\ 1 \end{bmatrix} \xrightarrow{\varphi} [y_1^0] = [\varphi(v_1^0] \quad (56)$$

where, $w_{ji}$ represents the synaptic weight connecting the output of neuron *i* to the input of neuron *j*, and *y* represents the outputs.

The error function or the cost function is used to measure the distance between the target and the output of the network. The weights of the network are updated in the direction that makes the error function minimum.

In the feedback pass, gradient-descent optimisation is used to minimise the output error, where,

$$e = \hat{y}_i - y_i \quad (57)$$

and,

$$E = \frac{1}{2}\sum e^2 \quad (58)$$

where $\hat{y}_i$ is the desired output of the network and $y_i$ is the actual output that the network has

been provided with. The aim of the training algorithm is to minimize *E*, by using a gradient descent is used to update the weights.

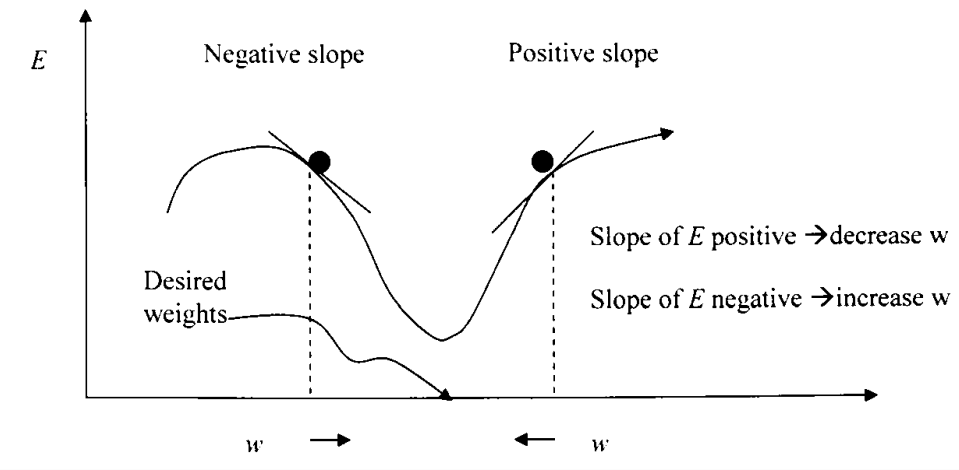



**Figure 43 Basic principle of gradient descent**

Figure 43 shows the behaviour of error *E* with respect to one weight *w*. In order to decrease the value of the error function *E*, the Back-propagation algorithm performs gradient descent in the reverse direction of the error gradient (slope). If the gradient of *E* is negative, *w* must be increased to move forward towards the minimum. If *E* is positive, *w* must decrease to move backward towards the minimum. By repeating this process, *E* 'travels downhill' until a minimum is reached.

The local gradient $\delta j$ for a hidden neuron *j* is given by,

$$[\delta_j^0] \widehat{=} [e][\varphi(v_j^0] = [e\varphi(v_j^0)] \quad (59)$$

and the weight corrections are calculated using,

$$w_{1x4}(k+1) = w_{1x4}^0(k) + \eta[\delta_1^0][y_1^2 \quad \dots \quad y_3^2 \quad 1] \quad (60)$$

$$\begin{bmatrix}\delta_1^2\\ \vdots\\ \delta_3^2\end{bmatrix} = \left\{\begin{bmatrix}w_{1,1}^0\\ \vdots\\ w_{1,3}^0\end{bmatrix}[\delta_1^0]\right\}\begin{bmatrix}\varphi(v_1^2)\\ \vdots\\ \varphi(v_3^2)\end{bmatrix} \quad (61)$$

$$w_{3x7}(k+1) = w_{3x7}^2(k) + \eta\begin{bmatrix}\delta_1^2\\ \vdots\\ \delta_3^2\end{bmatrix}[y_1^1 \quad \dots \quad y_6^2 \quad 1] \quad (62)$$

$$\begin{bmatrix}\delta_1^1\\ \vdots\\ \delta_6^1\end{bmatrix} \widehat{=} \left\{\begin{bmatrix}w_{1,1}^2 & \dots & w_{3,1}^2\\ \vdots & \ddots & \vdots\\ w_{1,6}^2 & \dots & w_{3,6}^2\end{bmatrix}\begin{bmatrix}\delta_1^2\\ \vdots\\ \delta_3^2\end{bmatrix}\right\}\begin{bmatrix}\varphi(v_1^1)\\ \vdots\\ \varphi(v_6^1)\end{bmatrix} \quad (63)$$

$$w_{6x5}(k+1) = w_{6x5}^1(k) + \eta\begin{bmatrix}\delta_1^1\\ \vdots\\ \delta_6^1\end{bmatrix}[x_1 \quad \dots \quad x_5 \quad 1] \quad (64)$$

## 7.3Process of learning MLP algorithm

Using the MLP model, training is carried out using all available data until the prediction error is minimized to $\leq 10^{-5}$. Firstly, optimum weights in the feed-forward part of the network are found, and then tested in the backpropagation part. If a large error emerges as a result of back-propagation, the algorithm is retrained to provide new weights. Prediction is carried out by using only the feedforward part of MLP.

The process of training the MLP algorithm is shown in Figure 44. Algorithm performance is achieved in five steps, as follow:

1. The network is initialized by inputting the generated training pairs
2. Initialize weights, $w$, at random
3. Select an appropriate error function ($e$) and sum of the square are calculated ($SSE$), defined in this thesis by the expression (57 and 38, respectively).
4. Apply the weighted change, $\Delta w$ to each weight, $w$ for each training pattern- with maximum of 1000 epochs. (Note: One set of updates for all the weights for all the training patterns is

called one epoch of training.) Repeat this step until the network error function is small enough or until all 1000 epochs are executed, whichever one comes first.

5. Apply the feedforward test to generate the predicted output

The first 133 days of data are as the initial training data set. This training set will predict the $147^{th}$ day as the $140^{th}$ day is used as a test set to predict 7 days ahead. This highlights an issue that must be accommodated because when the independent variables used as predictors by the model are calculated, the application almost always looks back in history and defines the predictors as functions of recent data. In addition, when the algorithm calculates the dependent variable that is predicted, it typically looks ahead and bases the variable on future values of the time series. At the boundary between a training period and a test period, these two regions can overlap, resulting in what is commonly termed future leak. In effect, future test-period behaviour of the time series leaks into the training data, providing information to the training and test procedures that would not be available in real life [114]. In this case, to prevent future leak, 7 data points are ignored.

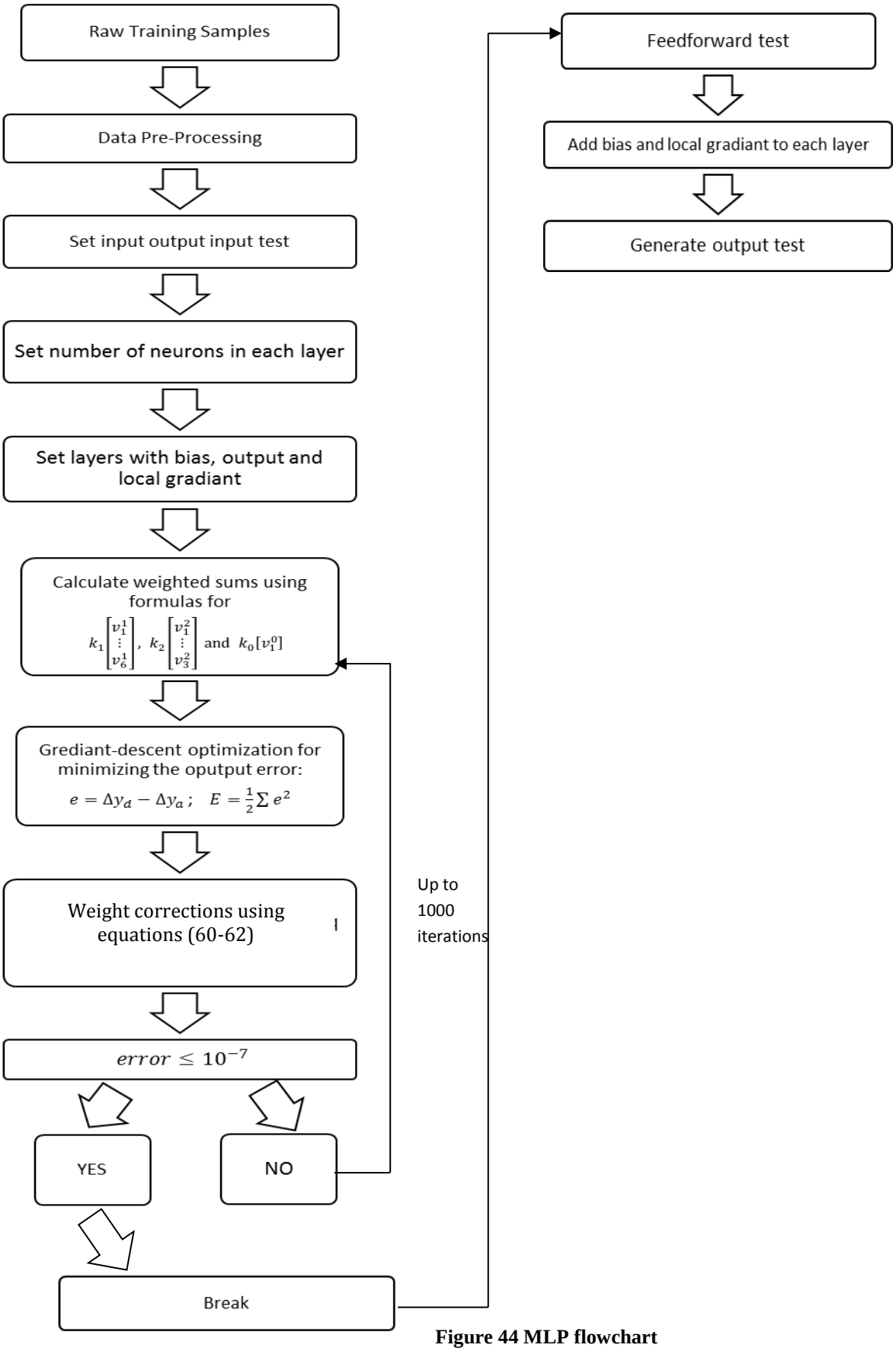



**Figure 44 MLP flowchart**

## 7.4 Results of MLP

### 7.4.1 MLP prediction results

Figure 45 shows the neutrophil count prediction results, with the performance accuracy summarised in Table IX.

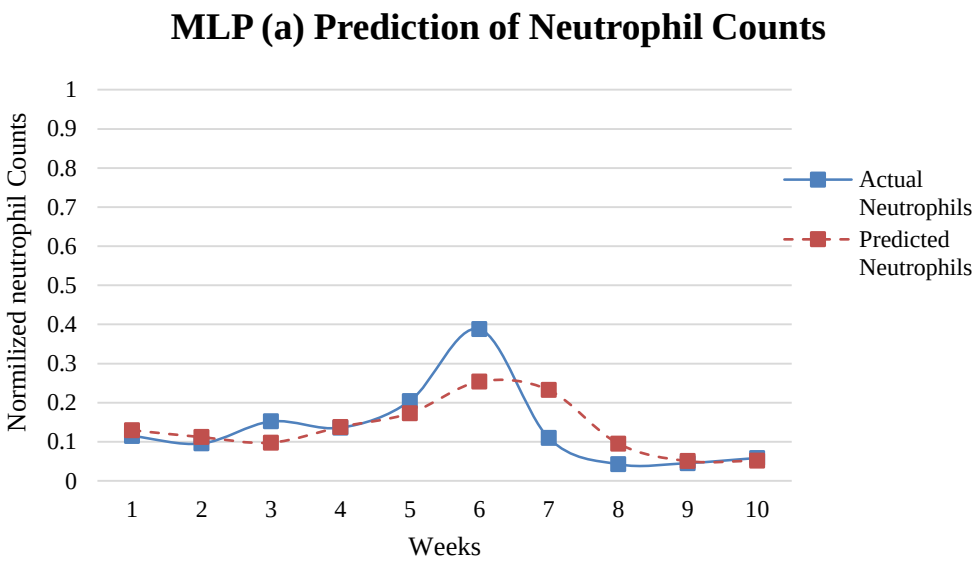



**Figure 45 Normalized neutrophil counts prediction (a) using MLP NN. Prediction period: 7 days ahead**

Considering the percentage error, the accuracy of MLP in this instance is, on average, over 60%. Data set (b) is used for the purpose of validating the algorithm again. The same model is on the chronologically later data and the results are shown in Figure 46.

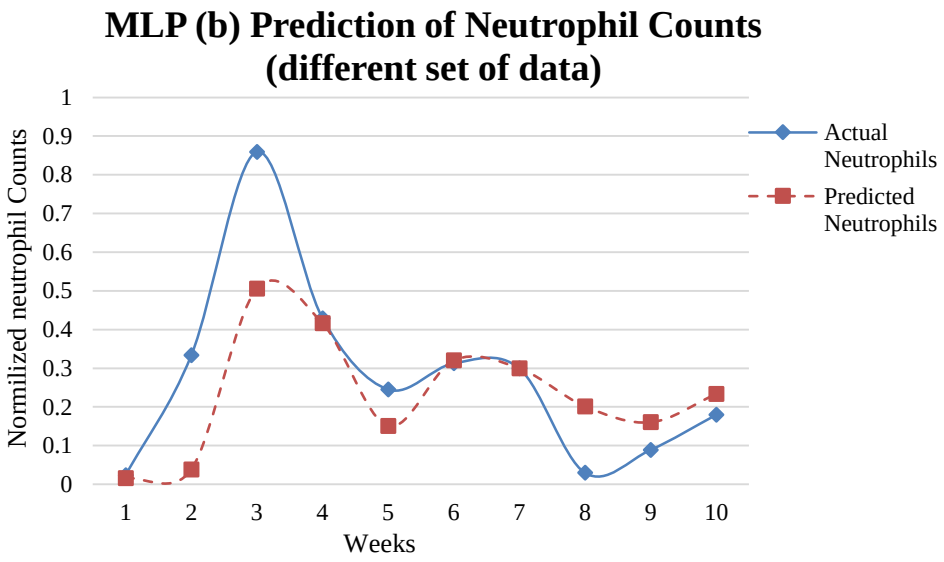



**Figure 46 Normalized neutrophil counts prediction (b) using MLP NN. Prediction period: 7 days ahead**

Finally, the data were used with a MLP model from the neural networks toolbox in Matlab—considered as case c). It can be seen from Figure 47 that performance of the matlab variant is very poor, as is also evident from the results of Table X and Table XI.

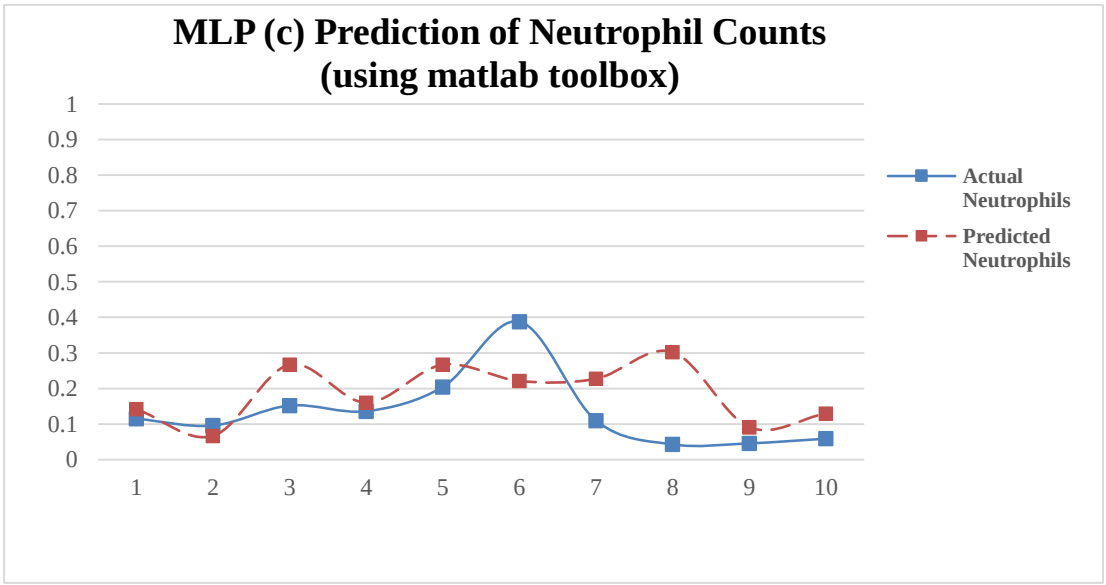



**Figure 47 Normalized neutrophil counts prediction (c) using MLP NN. Prediction period: 7 days ahead**

**Table X Error calculation for MLP models**

| Weeks<br>For (a) | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **Mean** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **APE** | 12.73 | 17.08 | 35.60 | 1.49 | 15.25 | 34.59 | 112.2 | 123.9 | 11.02 | 11.78 | 37.5 |
| **SSE** | 0.0002 | 0.0003 | 0.0029 | 4E-6 | 0.0009 | 0.0180 | 0.0151 | 0.0028 | 2E-5 | 4E-5 | 0.0040 |
| **For (b)** | | | | | | | | | | | |
| **APE** | 34.17 | 88.44 | 41.12 | 3.029 | 38.63 | 2.462 | 0.167 | 578.1 | 79.91 | 29.95 | 89.6 |
| **SSE** | 0.0001 | 0.0868 | 0.1249 | 0.0002 | 0.0089 | 5E-5 | 2E-7 | 0.0295 | 0.0051 | 0.0029 | 0.02584 |
| **For (c)** | | | | | | | | | | | |
| **APE** | 23.53 | 30.35 | 75.73 | 17.77 | 30.63 | 43.04 | 107.6 | 610.5 | 99.55 | 121.2 | 116 |
| **SSE** | 0.0007 | 0.0008 | 0.0132 | 0.0005 | 0.0039 | 0.0279 | 0.0139 | 0.0675 | 0.0021 | 0.0051 | 0.013 |

**Table XI Error calculation for MLP models**

| | ***RMSE*** | ***MSE*** | ***MAE*** | ***R^2*** | ***D*** |
|---|---|---|---|---|---|
| **MLP (a)** | 0.0440 | 0.0040 | 0.0440 | 0.9866 | 0.8356 |
| **MLP (b)** | 0.1069 | 0.0258 | 1.0688 | 0.5197 | 0.8082 |
| **MLP (c)** | 0.0910 | 0.0135 | 0.0919 | -0.4422 | 0.5431 |

Even though in some cases MLP (b) errors are worse than MLP (c). it is clear by the graphs and by $R^2$ and D that this is not the case. The MLP (b) that is the same as MLP (a) just tested on different part of data, does have a larger MAPE but this is only the case because MAPE does let the outlier dominate the final mean error. It is also clear that the MLP a and b do outperform the toolbox MLP (c). Since this is the best performing algorithm so far, it will be taken forward to modify it to real-time usage.

### 7.4.2 Confidence interval for MLP

The confidence interval is determined using the same procedure as for the previous SVM prediction case.

The standard error is calculated and used to produce 95% confidence interval (CI), Table XII and Figure 48. From this analysis it can be concluded that both MLP predictions are good since both of their mean points lie near the zero axes and the upper and lower boundaries are not far apart.

**Table XII Confidence interval for MLP prediction**

| | **Error in Neutrophil prediction and 95% confidence intervals** | | |
|---|---|---|---|
| ***Prediction algorithm*** | ***Lower*** | ***Mean ($10^9$ / L)*** | ***Upper*** |
| MLP (a) | 0.01561 | 0.04440 | 0.07239 |
| MLP (b) | 0.03448 | 0.10688 | 0.18131 |

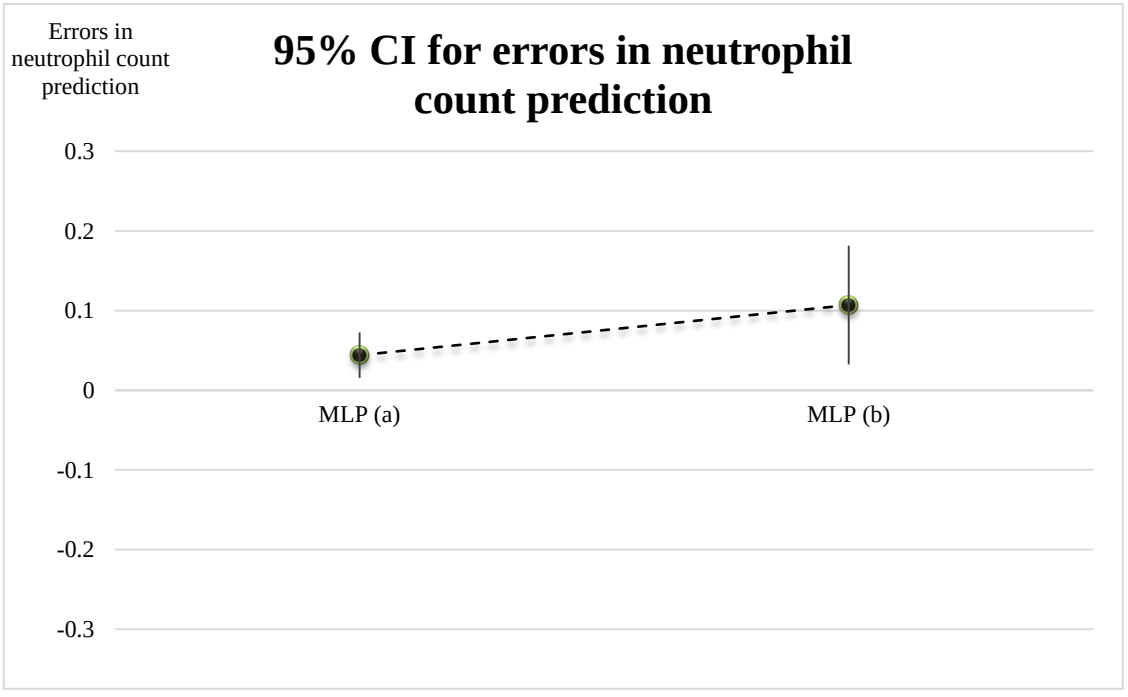



**Figure 48 Confidence interval for MLP prediction**

## 7.5Summary for MLP prediction

The results show predictions of neutrophil counts over successive 7-day horizons, resulting in prediction accuracies of up to 63%. The study shows that the MLP model produces marginally greater overall prediction accuracy compared to SVR counterparts. Further research to improve the algorithm and incorporate adaptive features into the developed MLP model, with a view to further improving prediction accuracy, is given in the next chapter.

# Chapter VIII – Continuous (incremental) learning multi-step prediction for neutrophil counts

## 8.1 Introduction

Machine learning algorithms can be classified into batch and continuous learning algorithms. Batch learning as described at the beginning of this thesis, is used when full data is available, and the prediction is made using the whole data set. Batch learning in the cases considered here changes the system weights as each 7 days of data become available. Online methods are those that "process one datum at a time". Continuous learning updates the weights whenever new data becomes available. Informally, non-incremental learning assumes that the world is closed. Although this can be theoretically convenient, it certainly does not hold in everyday life, where information, however much of it is available, is generally uncertain, incomplete and comes in continuously rather than all of it being available. If all the data relevant to the problem at hand is indeed available a-priori, then the world may be assumed closed. Otherwise, there is a need for special learning mechanisms that invalidate portions of knowledge, while not affecting the rest. Off-line learning such as the one mentioned, attempts to minimize the error and once the training cycle is completed, the network is put into operation. No further learning is permitted when the network is in the operating mode in order to preserve the learning knowledge base. This was the assumption for the previous investigations in Chapters 4-7, where batch data has been provided. However, in real life this will not be the case. The biggest drawback of off-line learning is that when the network is presented with previously unseen data samples, there is no built-in mechanism for the network to absorb new information into its knowledge base. To accommodate new information, the network needs to be retrained using the new data sample together with previous samples.

A word of caution is in order before discussing earlier work on continuous learning. This is because the phrase "continues learning" has been used rather loosely with widely different meanings in the pattern recognition and artificial learning literature, where the term refers to

diverse concepts such as incremental network growing, on-line learning, constructive learning, lifelong learning, and evolutionary learning.

The fact that the data considered in this research, and in other normal environments, will actually be updated periodically, means that there is a demand for developing a continuous sequence learning methodology.

It has long been argued that self-adaptation is a prerequisite for general intelligence and that learning, in particular, involves the ability to improve performance over time. Clearly, humans acquire knowledge over time, i.e., incrementally, since all of the information necessary to learn many concepts is rarely available a-priori. Rather, new pieces of information become available over time, and knowledge is constantly revised (i.e., evolves) based on newly acquired information [117].

In this chapter the previously proposed MLP is extended to be capable of dealing with new incoming data without retraining using the whole set of previous training data. It is shown that the resulting continuous MLP achieves better prediction accuracy compared to the batch learning MLP variant presented in the previous chapter.

A data streaming forecasting system for regression with continuous data streams, is developed.

A key difficulty in dealing with time series representing blood counts is their non-stationery characteristics, i.e. the underlying data production mechanism (age of patient for example) changes over time and that new data is only available once per week.

In summary, a continuous learning algorithm should possess the following main three properties:

1. It does not require learning from the start
2. A large scale forgetting of previous data does not occur once new data is learned

3. It is able to obtain additional information from the new data to improve its performance.

## 8.2 Continuous learning Neural Network Design

The use of a continuous data multi-layer perceptron technique is now presented for predicting neutrophil counts. Weights of the network are updated after processing each unit of the training data. Subsequently, updated weights are used to process the next unit of training data.

The encouraging results of experiments from the previous chapters provide a motivation to apply this type of modelling with the developed MLP neural network since it outperformed both the SVR and MATLAB-based MLP model variants.

The aim of the proposed algorithm is to learn from new data without losing prior knowledge. It modifies the weights of the MLP neural network architecture. The algorithm involves "growing" the network incrementally using the new data without requiring re-training using old data.

A flowchart showing the formulation of the proposed algorithm is shown in Figure 49. The addition of new training data to an already trained MLP incurs the following requirements:

- The trained MLP model
- Weights and biases of MLP
- MLP parameters
- New sample

The MLP then performs an update of the weights and bias while adhering to the constraints set initially, and returns the following outputs:

- New trained MLP model
- Updated weights

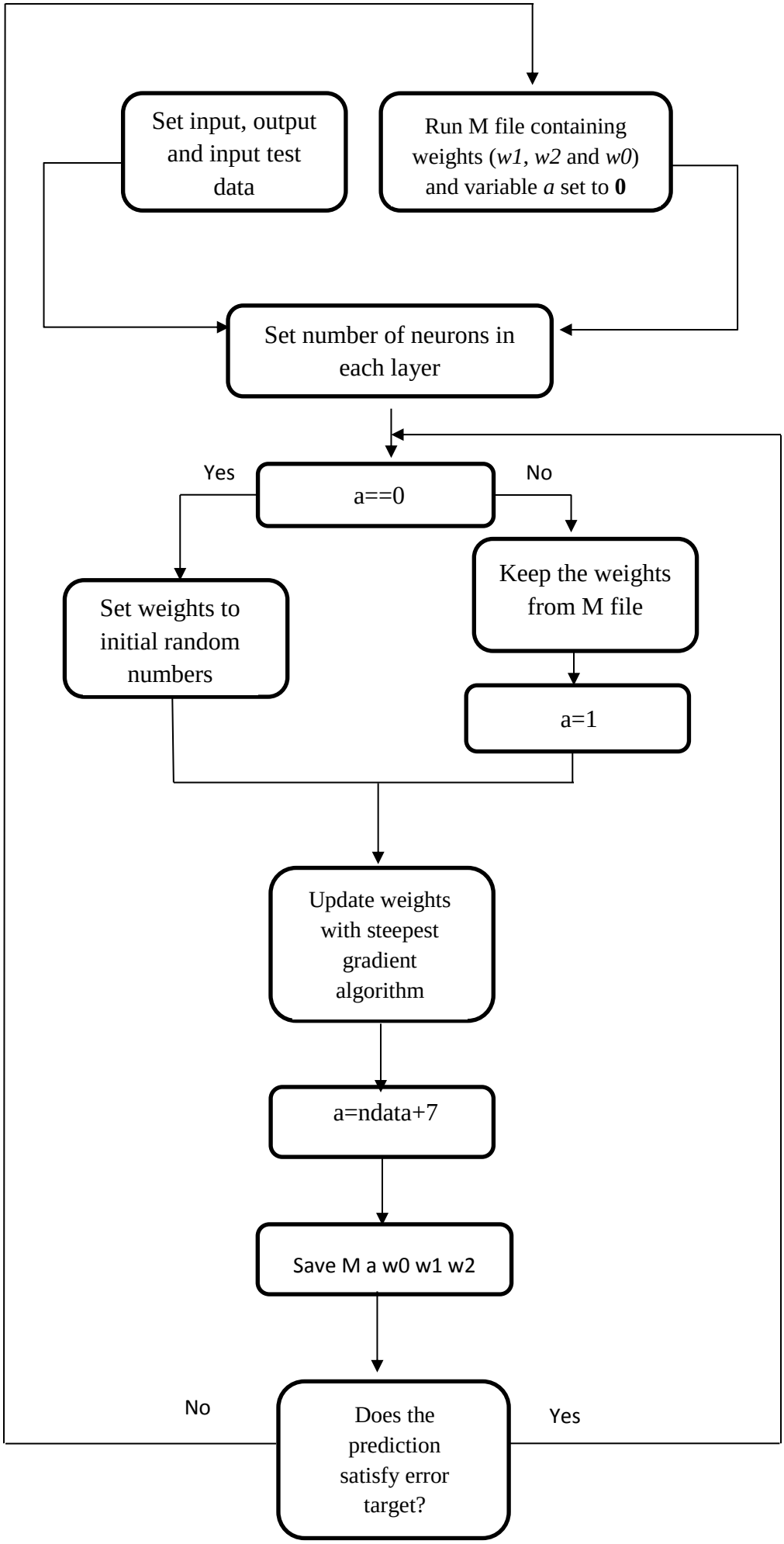



**Figure 49 Continuous MLP flowchart**

To satisfy the above requirements, the algorithm must execute the following tasks, (i) add new knowledge from incoming label data throughout time and (ii) predict the output of all new data inputs.

Task (i) refers to updating the weight parameters; in the proposed algorithm the principal graph is updated each time a new 7 days of data is presented. The updating procedure consists of introducing a new M.mat (seen bellow) file with a variable 'a' that represents the start of the counter (beginning at 0). Each time a set of 7 new data points arrive, the counter increases by 7. Task (ii) corresponds to predicting new data. In the proposed algorithm this task is performed by updating the M.mat file as the M.mat file also includes three variables of all three weights for the hidden layers that also start from 0 and every time the MLP is run with a prediction made, it remembers the new weight values of previous calls. The M.mat file loads in each new call with saved data from the previous call and finally rewrites the new results.

%M.mat

>> a=0; w1=0; w2=0; w0=0;

>> save M a w1 w2 w0

## 8.3 Prediction results and discussion

This section presents results from the proposed incremental algorithm. The same data set as in previous chapters, is used. The algorithm begins training using just 14 data points with one unlabelled datum point that is actually the 21$^{st}$ day. Every additional data set has seven new data points plus one unlabelled datum point.

In the following example, the data sets are presented one at a time in an interposed manner, i.e. every labelled set has one unlabelled one. The results are represented by the error percentage in predicting the unlabelled datum.

Initial training begins with random parameters, and for each generation, each individual has new random initial weights drawn. The aim is to evolve the various network parameters to produce networks with better incremental learning abilities.

The value of being able to forward predict online can be seen in Figure 50.

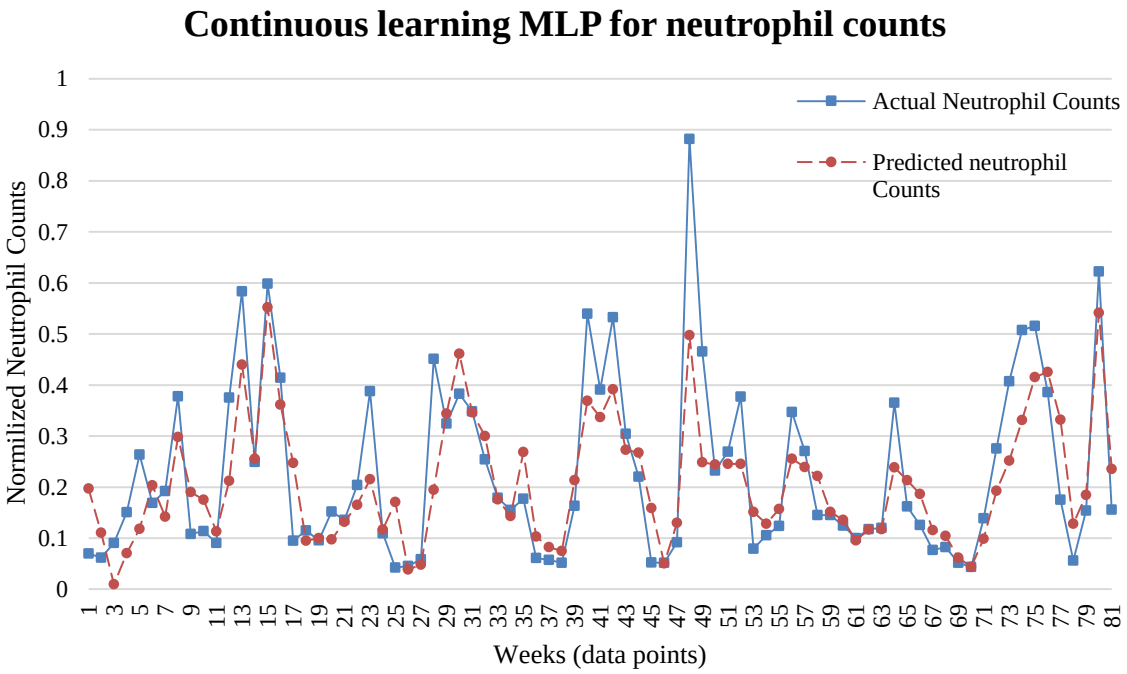



**Figure 50 Incremental MLP prediction**

From Figure 50, it can be seen that the prediction is generally very good as it follows the underlying characteristic well on the whole. Nevertheless, low prediction accuracy can be seen at some points, such as week 48. This could be due to underlying data quality—missing values were not considered problematic since normalisation and interpolation, from earlier were used in order to effectively fill the data that is missing. The data set had mostly weekly data except when patient was neutropenic. In this period blood was taken every day. In order to have the

whole data set as daily points, interpolation is used. However, some missing data could have led to inaccuracies propagating. Moreover, at this stage, no consideration of the effects of outliers [118] has been made.

A summary of comparative performance of the original batch MLP, with the newly proposed incremental updating variant, is shown in Figure 51, for weeks 1 to 10. Only during weeks 2 and 7 does the incremental variant outperform the original MLP.

Obtaining predictions from both models using data between weeks 40 to 50 provides the results shown in Figure 52. In this case it is clear that the incremental variant significantly outperforms the batch processing variant. Only weeks 4 and 10 are identified otherwise, albeit in week 4 the difference is only 2%.

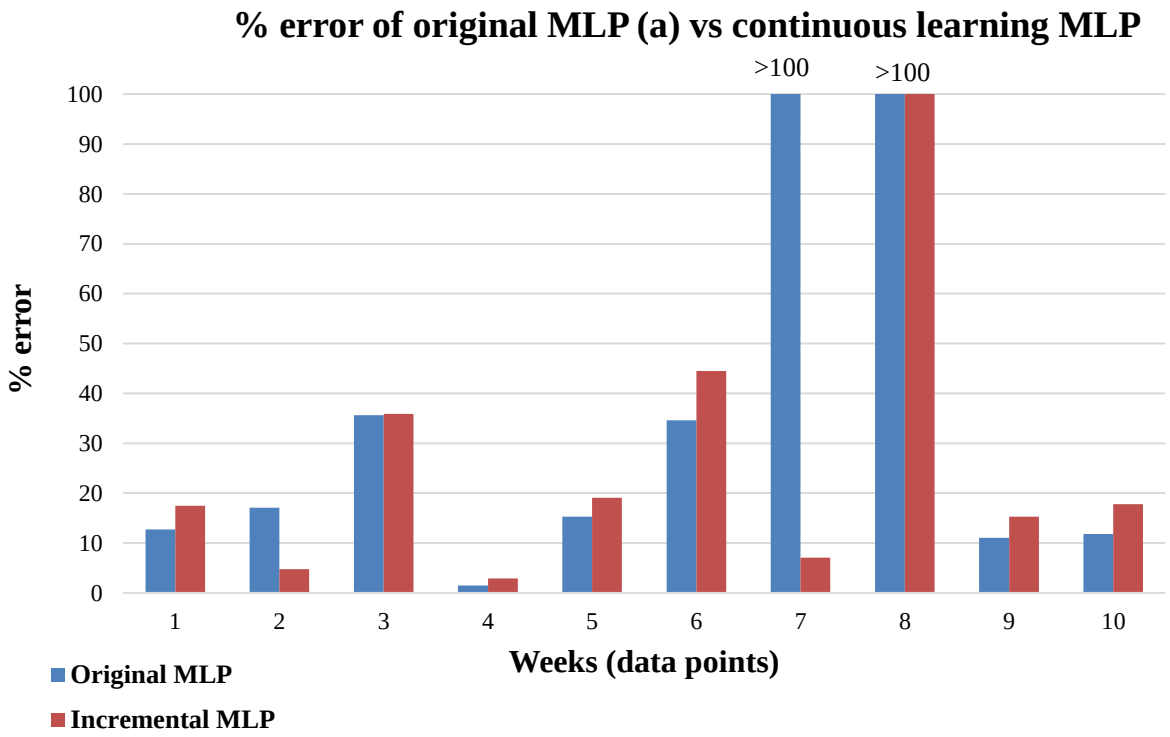



**Figure 51 Percentage error original batch learning MLP (a) vs continuous MLP**

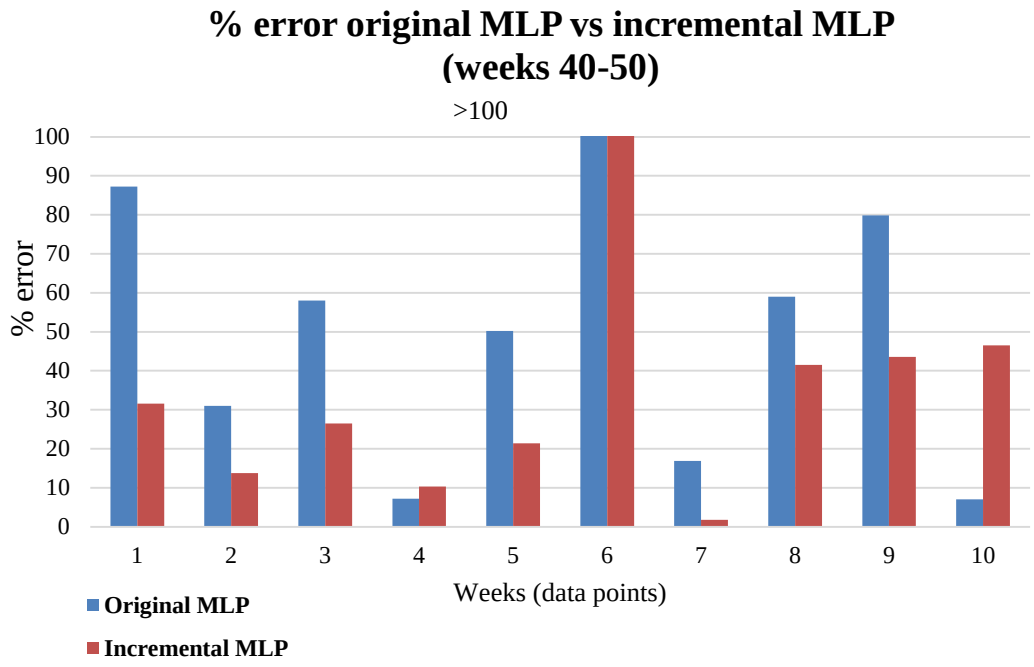



**Figure 52 Percentage error original batch learning MLP (b) vs continuous learning MLP**

Success of the model is also looked at from a different prospective because this model is only used on one data set and 100% accurate prediction is an unrealistic expectation by the very nature of the model. To relax the model's fitting requirement might be a good idea as in the data the dependent variable can occasionally have unusual values, and these unusual values might be due to heavy-tailed noise. And in this case, a single poor prediction can cause considerable damage. A standard performance indicator, is called the success factor. This is computed by dividing the sum of the successes by the sum of the failures, as shown in Equation (65) [114].

$$success\ factor = \frac{\sum_{x_i>0} x_i}{-\sum_{x_i<0} x_i} \qquad (65)$$

Using the Equation the success factor is given as:

$$success\ factor = \frac{\sum_{x_i>0} x_i}{-\sum_{x_i<0} x_i} = \frac{47}{34} = 1.38$$

However, what is a success and what is failure in our prediction. In the above calculation, 70% and above prediction accuracy is used as a success and anything below that is a failure, even if it's 1 or 2% less than 70%. Problem with the online MLP, seven out of 10 first predictions are much lower than the 70% accuracy mark. If first 10 predictions are taken out a success factor of 6.30 is obtained. A bigger number shows better success factor. 6.30 says that in every 6 good prediction only one bad prediction is made.

When this performance criteria is used, it is often important that some additional minimal performance constraint be imposed, such as the ones seen previously and not only rely on the success factor. Nevertheless, it is useful to put it into this kind of, different, perspective.

## 8.4 Summary

While learning in neural networks can be either off-line (batch) or online (continuous), online learning neural networks are beneficial to undertaking problems in non-stationary environments such as the weekly blood data stream. In this chapter, the online (continuous) learning implementation into the MLP network is highlighted and their ability how to solve problems is discussed.

A continuous learning system started from scratch and gained knowledge from examples given one at a time over time. As a result, the quality of its predictions improved over time. Characteristic in this is the fact that the system is not very trustworthy early on. Hence, although the system can make predictions at any time, one must be cautious with the predictions earlier on. Furthermore, it is hard to judge at what point has the continuous learning learned enough.

The feasibility of the approach has been validated on one real-world databases of a female leukaemia patient. It shows very promising results in learning new data. New parameters are implemented to a MLP as the base algorithm.

The algorithm has additional desirable qualities. Which are:

- It is intuitive and simple to implement
- It is applicable to a diverse set of neutrophil counts data and other input blood constituents.
- It can be trained very quickly.

However continuous learning has some disadvantages. The main drawback of this algorithm are:

- Frequent model updated can cause unwanted model output fluctuations
- models are generally more difficult to maintain, as online models can become unstable or corrupted due to contaminated data
- Need for continually monitoring of data and model quality
- Longer time to do the initial training with appropriate prediction
- Seems to have difficulties with unknown data more than an offline model.

# Chapter IX – Conclusion and further work

## 9.1 Introduction

This chapter summaries the final conclusions that can be derived from this research investigation. The novel contribution of this research work and possible research directions for use in the future are discussed.

## 9.2 Main conclusion derived from this research study

A relative appraisal of alternative candidate algorithms is given, and their performance compared for determining future neutrophil counts. Their strengths and capabilities for input-outputs modelling have been discussed. One candidate technique came as the most computationally efficient and capable of complex nonlinear modelling of data from a female ALL patient—the multi-layer perceptron, with some tuning of the parameters in the network. The network has been trained and tested on 10 sets of time series data points using 4 months of training data.

This thesis explained the fundaments of neutrophil count prediction, addressing the difficulties in neural networks and traditional forecasting approaches, particularly nonlinear prediction. A review of literature detailing the practical applications of many algorithms for time series prediction is given. Attention was given to pre-processing methods to reduce the number of blood constituent variables used for predictions. The design of neural networks to successfully predict non-linear time series has been presented. The methodology of constructing the network prediction model, as well as performance measures, has also been discussed. After data pre-processing, a detailed appraisal of two candidate algorithms was undertaken, and their performance compared. Finally, the technique showing the best performance, the MLP, was extended so that it can adaptively accommodate incremental learning when new data arrives. This online network has been trained and tested on 225 days of data from one female patience and shows improving accuracy as more data becomes available to learn from.

## 9.3 Research contribution

Despite being successfully applied in signal prediction, pattern recognition, data classification, function approximation and financial time series prediction, there was no research being carried out on neutrophil, on in that matter, any medical support prediction, using MLP or any other neural network. The main contributions drawn from this research are:

- Data pre-processing was done using normalization and scaling in order to increase the data points
- Feature selection of input data was carried out using PCA, Fishers score and backward elimination, which showed the main 4 inputs that contribute to an accurate neutrophil count prediction the most viz. White blood cells, Neutrophils day before, 6-MP and Platelets
- Application of NARX algorithm for one day ahead prediction
- Comparison of 20 algorithms was carried out finding the best two for this kind of data- MLP and SVM
- Batch learning of MLP and SVM to predict 7 days ahead prediction with 63% accuracy was computed using MATLAB
- MLP algorithm was modified to continuously learn the data as it comes making it a more real-time prediction.

Given the current findings, the study should provide significant motivation for further research as it suggests that algorithms can be effectively used to better inform medical forecasting.

## 9.4 Future work

Due to limitations of experiments, more tests and research is required to define and quantitatively evaluate the performance of the algorithms.

The method which was proposed and presented in this research work can be viewed as starting points for future research direction, since the potential of algorithms predicting neutrophil counts is by far not exploited yet. More research is needed with the use of MLP and online MLP to give more accurate results in terms of predicting human cell count in order to moderate the intake of drugs more efficiently. Based on the conclusions of this thesis and the lack of other research in this area, the following continuations of this research work are suggested:

- **More data –** it is obvious, the more data the algorithm has the better it will perform. The concept proves high possibility of MLP being able to predict neutrophils in childhood cancer in order for the patients not to reach neutropenia. However, this is only proven on one female patient's data, and it would be highly recommended to gather more data to train the algorithms further. Also, in our study the assumption is made that all the information needed in order to predict the neutrophil counts time series is included in the input-output series. But is this assumption valid or are there other forms of input data that can offer extra information to our model? Such as age of the patient, sex of the patient etc.
- **Algorithms and noise reduction**

  The algorithm comparison could be done in more detail. In order to be certain that they cannot find patterns in the neutrophil counts time series the case of tracing patterns has to be examined in smaller time periods than the ones used and also use it on different parts of data. However, this was not feasible in the time frame of the project. However, as the SVR experiment shown, the algorithm preforms well in one part of data and very poorly in the other part- this could have been a case of some other algorithms.

  It is also considered that the daily patients' blood data is highly noisy data. Therefore, it is believed that in future studies there needs to be a way of reducing the noise of the data the model is been fed with. One way in which this can be achieved is by moving

from daily data to weekly or monthly data, this way the trends that exist in the data would be clearer and thus easier to be traced by the prediction models and no assumption would be made by interpolating the data. This again can only be achieved if there was more data.

- **Optimization**- one of the options would be to optimize the current algorithm. This might be possible for the original MLP and the online MLP. Algorithms such as genetic algorithms are most probably the easiest ones to implement into neural network architecture. Genetic Algorithms (GAs) are search and optimization techniques based on the stochastic approach enhanced by the principles of biological evolution in nature [119]. The GA is a well-known meta-heuristic algorithm, following the natural evolution processes. GAs work using the idea of chromosome, which encodes the genetic information of an individual. The chromosomes are evaluated on the objective function, which is the desired objective of the problem.

  However, GAs are not guaranteed to find the optimal solution like other meta-heuristic algorithms. The GA starts by defining optimization variables, objective functions and control algorithms [120]. They have been widely used in forecasting financial markets. In this case, they have been used to find the best combination value of parameters. The biggest advantage for this case - they can also be built into ANN models. GA is created mathematically using vectors [121].

  If genetic algorithm is to be used as an optimization procedure in the algorithm for prediction. The way this can be achieved is to firstly choose parameters to optimize and determine chromosomal representation of parameters. Then, fitness needs to be evaluated of each individual. Random behaviour and selection rules will select the next "population".

- **Hybrid-model –** Other way to make the prediction better is to introduce hybrid models.
- **Bio-modelling –** System biology can also be done in this research as it will show more of the medical side.
- **User friendly interface system-** Another task is to make the MATLAB codes more use friendly- create a user interface that will allow the medical professionals to only choose the algorithm and everything else runs in the background.

## 9.5 Summary

This research work underlines an important contribution of the proposed MLP algorithms: namely their elegant ability to approximate nonlinear time series. This superior property held by this algorithm could promise more powerful applications in many other real-world problems associated with drug delivery such as schizophrenia. To conclude, this research has started something new, using promising intelligent computational technology, in order to predict the drug dosage of one ALL patient. The batch learning MLP gives promising results but the prediction algorithm based on self-learning MLP neural network has higher predictive accuracy and is more stable.

# References


[1] "Leukemia," [Online]. Available: http://nihseniorhealth.gov/leukemia/whatisleukemia/01.html. [Accessed 29 September 2014].

[2] "Chilhood Cancer: Leukemia," Kids Health from Nemours, [Online]. Available: http://kidshealth.org/parent/medical/cancer/cancer_leukemia.html. [Accessed 3 October 2014].

[3] "American cancer society," 2 March 2014. [Online]. Available: http://www.cancer.org/cancer/leukemiainchildren/detailedguide/childhood-leukemia-treating-children-with-all. [Accessed 2 October 2014].

[4] "We Are Macmillan cancer support," 1 January 2014. [Online]. Available: http://www.macmillan.org.uk/Cancerinformation/Cancertypes/Leukaemiaacutelymphoblastic/TreatingALL/Chemotherapy.aspx. [Accessed 6 10 2014].

[5] "Chemo care," [Online]. Available: http://chemocare.com/chemotherapy/drug-info/6-mp.aspx. [Accessed 30 09 2014].

[6] J. E. Walsh, Handbook of Nonparametric Statistics, New York, 1962.

[7] W. J. Conover, Practical Nonparametric Statistics, New York: Wiley & Sons, 1980.

[8] D. Hosseinzadeh and S. Krishnan, "Gaussian mixture modeling of keystroke patterns for biometric applications," *IEE Trans. Syst.*, vol. 38, no. 6, pp. 816-826, 2008.

[9] G. Li, T. Y. Leong and L. Zhang, "Translation initiation sites prediction with mixture Gaussian models in human cDNA sequences," *IEEE Transaction on Knowledge and Data Engineering*, vol. 17, no. 8, pp. 1152-1160, 2005.

[10] T. Chen, J. Morris and E. Martin, "Probability density estimation via an infinite Gaussian mixture model: Application to Statistical Process Monitoring," *Journal of the Royal Statistical Society: Series C,* vol. 55, no. 5, pp. 699-715, 2006.

[11] H. Greenspan, A. Ruf and J. Goldberger, "Constyained Gaussian mixture model framework for automatic segmentation of MR brain images," *IEEE Transactions on Medical Imaging,* vol. 25, no. 9, pp. 1233-1245, 2006.

[12] P. Paalanen, J. K. Kamarainen, J. Ilonen and H. Kaelviaeinen, "Feature representation and discrimination based on Gaussian mixture model probability densities- Practices and algorithms," *Pattern Recognition,* vol. 39, no. 7, pp. 1346-1358, 2006.

[13] M. Ouyang, W. J. Welsh and P. Georgopoulos, "Gaussian mixture clustering and imputation of microarray data," *Bioinformatics,* vol. 20, no. 6, pp. 917-923, 2004.

[14] J. Clark, P. Kountouriotis and R. Vinter, "A Gaussian Mixture Filter for Range-Only Tracking," *IEEE,* vol. 56, no. 3, pp. 602-613, 2011.

[15] Y. Ozbek, M. Hasegawa-Johnson and M. Demirekler, "Estimation of Articulatory Trajectories Based on Gaussian Mixture Model (GMM) With Audio-Visual Information Fusion and Dynamic Kalman Smoothing," *IEEE,* vol. 19, no. 5, pp. 1180-1195, 2010.

[16] A. Nikseresht and M. Gelgon, "Gossip-Based Computation of a Gaussian Mixture Model for Distributed Multimedia Indexing," *IEEE Transactions of Multimedia,* vol. 10, no. 3, 2008.

[17] A. P. Dempster, N. Laird and D. B. Rubin, "Maximum likelihood from incomplate data via the EM algorithm," *Journal of the Royal Statistical Society: Series B,* vol. 39, no. 1, pp. 1-38, 1977.

[18] M. Omachi, S. Omachi, h. Aso and T. Saito, "Pattern recognition using boundery data of component distributions," *Elsevier, Computers & Industrial Engineering,* vol. 60, pp. 466-472, 2010.

[19] W. James and C. Stein, “Estimation with quadratic loss,” in *Proceedings of the fourth Berkeley symposium on mathematical statistics and probability*, California, 1961.

[20] M. Sakai, M. Yoneda and H. Hase, “A new robust quadratic discriminant function,” in *Proceeding of the 14th international conference on pattern recognition*, Brisbane, 1998.

[21] S. Omachi, M. Omachi and H. Aso, “An approximation method of the quadratic discriminant function and its application to estimation of high-dimensional distribution,” *IEICE Transactions on Information and Systems* , Vols. E90-D, no. 8, pp. 1160-1167, 2007.

[22] L. R. Bahl, F. jELINEK and r. mERCER, “A Maximum Likelihood Approach to Continuous Speech Recognition,” *IEEE Trans. Pattern Anal. and Mach. Intell*, Vols. PAMI-5, no. 2, pp. 179-190, 1983/2009.

[23] F. Jelinek, “Continous speec recognition by statistical methods,” *IEEE,* vol. 64, pp. 532-536, 1976.

[24] B. H. Juang and L. R. Rabiner, “Hideen Markov Model for Speec Recognition,” in *Technometric*, vol. 33, 1991, pp. 251-272.

[25] L. R. Rabiner, “A Tutorial on Hidden Markov Models and Selected Applications in Speech recognition,” *Proc. IEEE,* vol. 77, no. 2, pp. 257-286, 1989.

[26] P. Baldi, Y. Chauvin, T. Hunkapiller and M. A. McClure, “Hidden Markov models of biological primary sequence information,” *Proc. Natl. Acad. Sci. USA* , vol. 91, no. 3, pp. 1059-1063, 1994.

[27] S. R. Eddy, “Profile hidden Markov model,” *Oxfrod Journals, Bioinformatics,* vol. 14, no. 9, pp. 755-763, 1998.

[28] A. Siepel and D. Haussler, "Combining Phylogenetic and Hidden Markov Models in Biosequence Analysis," *Journal of Computational Biology*, vol. 11, no. 2-3, pp. 413-428, 2004.

[29] J. Li and R. M. Gray, Image Segmentation and Compression Using Hidden Markov Model, The Springer International Series in Engineering and Computer Science, 2000.

[30] H. Attias, "Inferring Parameters and Structure of Latent Variable Models by Variational Bayes," *Proc. 15th conf. on Uncertainty in Artificial Intelligence*, 1999.

[31] J. H. Kim and J. Peal, "A computational model for casual and diagnostic reasoning in inference systems," *Proc. 8th Inter. Conf. on Artificial Intelligence,* pp. 190-193, 1983.

[32] Z. Ghahramani, "An Introduction to Hidden Markov Models and Bayesian Networks," *International Journal of Pattern Recognition and Artificial Intelligence* , vol. 15, no. 1, pp. 9-42, 2001.

[33] S. Theodoridis and K. Koutroumbas, Pattern Recognition, Fourth Edition, Elsevier Inc., 2009.

[34] L. Harald, Topics on Applied Mathematical Statistics, Stockholm: KTH, 2013.

[35] P. Kennedy, A Guide to Econometrics, 6th Edition, Wiley-Blackwell, 2008.

[36] J. D. Hamilton, Time series analysis, volume 2, Princeton: Princeton university press, 1994.

[37] A. S. Weigend, Time series prediction: forecasting the future and understanding the past, Santa Fe Institute Studies in the Sciences of Complexity, 1994.

[38] J. T. Yao, C. L. Tan and H. L. .. Poh, "Neural networks for technical analysis: a study on KLC," *International Journal of Theoretical and Applied Finance*, vol. 2, no. 2, p. 221–241, 1999.

[39] J. V. Hansen, J. B. Mcdonald and R. D. Nelson, “ime series prediction with genetic-algorithm designed neural networks: an empirical comparison with modern statistical models,” *Computational Intelligence,* vol. 15, no. 3, pp. 171-184, 1999.

[40] S. R. A. Fisher, Statical Method for Research Workers, London, 1925.

[41] L. Madrigal, “Analysis of variances (ANOVA),” in *Statistics for Anthropology*, Cambridge University Press, 1998, pp. 113-129.

[42] M. P. Menden, F. Iorio, U. M. M. Garnett, C. H. Benes, P. J. Ballester and J. Seaz-Rodriquez, “Machine Learning Prediction of Cancer Cell Sensitivity to Drug Based on Genomic and Chemical Properties,” *Plos One,* 2013.

[43] L. G. C. P. A. S. G. G. A. B. Q. Y. J. L. Y. S. I. O. T. G. J. S.-L. S. N. R. D.B. Solit, “BRAF mutation predicts sensitivity to MEK inhibition,” *Nature*, vol. 439, no. 7074, pp. 358-362, 2006.

[44] “INVESTOPEDIA,” [Online]. Available: http://www.investopedia.com/terms/t/t-test.asp. [Accessed 08 December 2014].

[45] “The T-Test,” in *Essential statistics for public managers and policy analysts*, CQ Press, 2011, pp. 179-197.

[46] Y. Qin, Z. L. Yu, C. D. Wang, Z. Gu and Y. Li, “A Novel clustering method based on hybrid K-nearest-neighbor graph,” *Pattern recognition,* vol. 74, pp. 1-14, 2018.

[47] V. Vapnik, The Nature of the Statistical Learning Theory, New York: Springer-Verlag, 1995.

[48] N. Deng and Y. Tian, The New Approach in Data MiningSupport Vector Machines, Beijing : Science Press, 2004.

[49] Y. Shi, Y. Tian, G. Kou, Y. Peng and J. Li, Optimization Based Data Mining: Theory and Applications, Springer, 2011.

[50] H. Tamura and K. Tanno, "Midpoint validation method for support vector machines with margin adjustments technique," 2009.

[51] S. Barnhard and A. Smola, Learning with Kernels Support Vector Machines, regularization, Optimization and Beyond, Cambridge: The MIT Press, 2002.

[52] U. Thissen, R. v. Brakel, A. d. Weijer, W. J. Melssen and L. Buydens, "Using support vector machines for time series prediction," *Elsevier*, 2002.

[53] A. J. Smola and B. Scholkopf, "A tutorial on support vector regression," *Statistics and Computing*, vol. 14, no. 3, pp. 199-222, 2004.

[54] C. L. P. S. T. Raicharoen, "Application of critical support vector machine to time series prediction," *Circuits and Systems*, vol. 5, pp. 741-744, 2003.

[55] J. S. a. J. Vandewalle, "Least squares support vector machines classifiers," *Neural Processing Letters*, vol. 9, no. 3, pp. 293-300, 1999.

[56] J. S. a. J. Vandewalle, "Recurrent least squares support vector machines," *IEEE Trans. Circuits Systems-I*, vol. 47, no. 7, pp. 1109 - 1114, 2000.

[57] J. Suykens and J. Vandewalle, "Least Swuares Support Vector Machine Classifiers," *Neural Processing Letters*, vol. 9, no. 3, pp. 293-300, 1999.

[58] I. Steinwart and A. Christmann, Suppor Vector Machines, Springer, 2008.

[59] J. A. Suykens, L. L. J. De Brabanter and J. Vandewalle, "Weighted least squares support vector machines: robustness and sparse approximation," *Neurocomputing*, vol. 48, pp. 85-105, 2002.

[60] C. M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.

[61] D. G. Tzikas, L. Wei, A. Likas, Y. Yang and N. P. Galatsanos, "A TUTORIAL ON RELEVANCE VECTOR MACHINES FOR REGRESSION AND CLASSIFICATION WITH APPLICATIONS," *EURASIP News Letter*, vol. 17, no. 2, 2006.

[62] "Saedsayad Decision Tree- Regression," [Online]. Available: https://www.saedsayad.com/decision_tree_reg.htm. [Accessed 14 October 2017].

[63] E. Lundkvist, "Decision Tree Classification and Forecasting of Pricing Time Series Data," July 2014. [Online]. Available: http://www.diva-portal.org/smash/get/diva2:746332/FULLTEXT01.pdf. [Accessed 06 September 2017].

[64] K. P. Murphy, "An Introduction to Graphical Models on CiteSeer," 10 May 2001. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=B51192D80591DB85F5F70D20E2F30435?doi=10.1.1.29.1410&rep=rep1&type=pdf. [Accessed 10 September 2018].

[65] W. S. McCulloch and W. Pitts, "A Logical Calculus of The Ideas Immanent in Nervous Activity," *Bull. Math. biophys.*, vol. 5, pp. 155-133, 1943.

[66] E. Collins, S. Ghosh and C. Scofield, DARPA Neural Network Study, AFCEA International Press, Fairfax, VA, 1988.

[67] Haykin, Neural Networks. A Comprehensive Foundantion, New York, NY: Macmillan, 1994.

[68] A. Zaknich, Neural Networks for Intelligent Signal Processing, World Scientific , 2003.

[69] "CS231n Convolutional Neural Networks for Visual Recognition," [Online]. Available: http://cs231n.github.io/neural-networks-1/. [Accessed September 2018].

[70] G. M. Maggiora, D. W. Elrod and R. Trenary, “Computational neural networks as model free mapping device,” *J. Chem. Inf. Comp. Sci.,* vol. 32, pp. 732-741, 1992.

[71] J. Dayhoff and J. DeLeo, “Artificial neural networks;Opening the black box,” 2001, pp. 1615-1635.

[72] R. Beale and T. Jackson, Neural Computing: An Introduction, Philadelphia, PA: Hilger, 1991.

[73] D. Svozil, V. Kvasnichka and J. Pospichal, “Introduction to multi-layer fee-forward neural networks,” *Elsevier, Chem. and Int. Lab. Sys.,* vol. 39, pp. 43-62, 1997.

[74] M. Castilla, J. Alvarez, M. Ortega and M. Arahal, “Neural network and polynomial approximated thermal comfort models for HVAC systems,” in *building and Environment 59,* 107-115, 2013.

[75] P. Tenti, “Forecasting foreign exchange rates using recurrent neural networks,” *Applied Artificial Intelligence,* vol. 10, no. 6, pp. 567-582, 1996.

[76] C. W. Omlin and C. L. Giles, “Extraction of Rules from Discrete-time Recurrent Neural Networks,” *Neural Networks ,* vol. 9, no. 1, pp. 41-52, 1996.

[77] J. Moody, L. Wu, Y. Liao and M. Saffell, “Performance Function and Reinforcement Learning For Trading Systems and Portfolios,” *Journal of Forecasting,* vol. 17, pp. 441-470, 1998.

[78] M. I. Jordan, “Serial order: A parallel distributed processing approach,” Institute for Cognitive Science Report 8604, UC San Diego, 1986.

[79] J. L. Elman, “Finding Structure in Time,” *Cognitive Science,* vol. 14, pp. 179-211, 1990.

[80] V. R. Mankar and A. A. Ghatol, “Design of Adaptive Filter Using Jordan/Elman Neural Network in a Typical EMG Signal Noise Removal,” *Advances in Artificial Neural Systems,* 2009.

[81] J. J. Hopfield, “Neural networks and physical systems with emergent collective computational abilities,” *Proceedings of the National Academy of Sciences of the USA*, vol. 79, no. 8, p. 2554–2558, 1982..

[82] S. Sathasivam, N. Hamadneh and O. H. Choon, “Comparing Neural Networks: Hopfield Network and RBF Network,” *Applied Mathematical Sciences,* vol. 5, no. 69, pp. 3439 - 3452, 2011.

[83] Y. Becerikli, A. F. Konar and T. Samad, “Intelligent optimal control with dynamic neural networks,” *Neural Networks,* vol. 16, pp. 251-259, 2003.

[84] S. Chen, Q. Zhang and C. Wang, “Existance and stability of equilibria of the continuous-time Hopfields neural network,” *J. Comput. App. Math.,* vol. 169, pp. 117-125, 2004.

[85] C. J. A. Bastos-filho, W. H. Schuler, A. L. I. Oliveira and L. N. Vitorino, “Routing algorithm based on Swarm Intelligance and Hopfield Neural Network applied to communication networks,” *Ellectronics letters,* vol. 44, pp. 2777-2783, 2008.

[86] M. Sheikhan and E. Hammati, “High reliable disjoint path set selection in mobile ad-hoc network using hopfield neural network,” *IET Communication ,* vol. 5, pp. 1566-1576, 2011.

[87] T. Kohonen, “Self-Organisation of Topologically Correct Feature Maps,” *Biological Cybernetics,* vol. 43, pp. 59-69, 1982.

[88] M. Kubat, “Neural networks: a comprehensive foundation by simon haykin, macmillan,” *The Knowledge Engineering Review,* pp. 409-412, 1999.

[89] J. Wang, J. Wang, Z. Zhang and S. Guo, “Stock index forecasting based on a hybrid model,” *Omega Elsevier,* vol. 40, pp. 758-766, 2012.

[90] J. Jang, “ANFIS: adaptive-network-based fuzzy inference system,” *IEEE Trans. Syst. Man Cybern.* , vol. 23, pp. 665-685, 1993.

[91] R. QI and M. A. Brdys, “Adaptive Fuzzy Modelling and Control for Discrete-Time Nonlinear Uncertain Systems,” in *American Control Conference*, Portland, 2005.

[92] M. A. Boyacioglu and D. Avci, “An Adaptive Network-Based Fuzzy Inference System (ANFIS) for the prediction of stock market return: The case of the Istanbul Stock Exchange,” *Elsvier: Expert Systems with Applications,* vol. 37, pp. 7908-7912, 2010.

[93] K. Narendra and P. Gallman, “In iterative method for the identification of nonlinear systems using a Hammerstein model,” *IEEE Transactions on Automatic Control,* vol. 11, no. 7, p. 546–550, 1966.

[94] M. Schetzen, The Volterra and Wiener Theories of Nonlinear Systems, Wiley, 1980.

[95] R. Abbasi-Asl, R. Khorsandi, S. Farzampour and E. Zahedi, “Estimation of Muscle Force with EMG Signals Using Hammerstein-Wiener Model,” in *5th International Conference on Biomedical Engineering* , Kuala Lumpur, 2011.

[96] A. Will, T. B. Schon, L. Ljung and B. Ninness, “Identification of Hammerstein–WienerModels,” *Automatica,* vol. 49, no. 1, pp. 70-81, 2013.

[97] R. E. Kalman, “A new approach to linear filtering and prediction problems,” *Trans ASME J Basic Eng.* , vol. 82, no. 1, pp. 35-45, 1960.

[98] S. V. Kumar, “Traffic Flow Prediction using Kalman Filtering Technique,” in *10th International Scientific Conference Transbaltica 2017: Transportation Science and Technology*, 2017.

[99] J. Sun, “Examples of validating an adaptive kalman filter model for short-term traffic flow prediction,” in *Twelfth COTA International Conference of Transportation Professionals*, 2015.

[100] S. Chapra and R. Canale, Numerical Methods for Engineers, Sixth Edition, New York: McGraw-Hill, 2010.

[101] Jolliffe, Principal Component Analysis 2nd ed., Springer, 2002.

[102] M. Drew, G. R. H. Wilden, W. Spillane, R. M. Walsh, C. A. Ryder and J. M. Simmie, Agric. Food Chem., 1979, p. 213.

[103] R. Mannohold, G. Cruciani, K. Dross, R. RekkerJ and Comput, “Aided Mol. Design 12,” 1991.

[104] Seybold, “SQERED 10,” 1999.

[105] A. L. K. Heberger, “Org. Chem.,” 1998.

[106] K. Heberger and M. Gorgenyi, “Chromatogr.,” 1999.

[107] Damborsky, Berglund, Kuty, Ansorgova, Nagata and Sjostrom, “Quant. Struct.-Act. Relat.,” 1998.

[108] A. Zaliani and E. Gancia, “Inf. Comput. Sci.,” 1999.

[109] Rowland and Todd, “Orthogonal Transformation,” Wolfram MathWorld.

[110] S. Kara and F. Direngali, “A system to diagnose atherosclerosis via wavelet transforms, principle component analysis and artificial neural networks,” *Expert Systems with Applications,* vol. 32, pp. 632-640, 2007.

[111] F. Song, Z. guo and D. Mei, “feature selection using principal component analysis,” in *IEEE Int Conference on System Science, Engineering Design and manufacturing Informatization* , 2010.

[112] a. D. W. Y. Shachmurove, “Utilizing Artificial Neural Network model to predict stock markets,” *Shachmurove, Y. and Witkowska, D. (2000). UtilizingCARESS Working Paper, Series No. 00-11, Pennsylvania: University of Pennsylvania, Center for Analytic Research in Economics and the Social Sciences,* 2000.

[113] K. A. Palit and D. Popovic, “Neural Networks Approach,” in *Computational Intelligence in Time Series Forecasting*, London, Springer, 2005, pp. 79-142.

[114] T. Masters, Assessing and Improving Prediction and Classification, Springer, 2018.

[115] N. Cristianini, An Introduction to Support Vector Machines and Other Kernel-based Learning Methods, Cambridge University Press, 2000.

[116] S. Haykin, Neural Networks, A Comprehensive Foundation, Delhi: Pearson Education (Singapore) , 2005.

[117] H. Samon, “Why should machine learn?,” in *Machine learning: An artificial Intelligence Approach* , Palo Alto, CA, Tioga publishing Company, 1983.

[118] J. Han and K. Kamber, “Data mining: concepts and techniques,” in *Academic Press*, San Francisco, 2001.

[119] D. Goldberg, “Genetic Algorithms,” in *Search. Optimization and MAshine Learning, 1st Edition,* Boston, MA, USA, Addison-Wesley Longman Publishing Co., Inc., 1989.

[120] A. Sokolov and D. Whitley, “Unbiased tournament selection,” in *The Conference on Genetic and Evolutionary Computation* , Washington, 2005, pp. 1131-1138.

[121] J. Kuepper, “Using genetic Algorithms To Forecast Financial Markets,” INVESTOPEDIA, [Online]. Available: http://www.investopedia.com/articles/financial-theory/11/using-genetic-algorithms-forecast-financial-markets.asp. [Accessed 27 October 2014].

[122] K. V. Mardia, J. Kent and J. M. Bibby, in *Multivariate Analysis*, London, Academic Press, 1979, p. 213.

[123] G. Kingston, Bayesian Artificial Neural Networks in Water Resources Engineering, Australia : School of Civil and Environmental Engineering, Faculty of Engineering, Computer and Mthematical Science, 2006.

[124] M. H. A.W. Minns, “Artificial neural networks as rainfall-runoff models,” *Hydrological Science Journal*, vol. 41, no. 3, 1996.

[125] G. D. H.R. Maier, “Neural networks for the prediction and forecasting of water resources variables: a review of modelling issues and applications,” *Environmental Modelling & Software* , vol. 15, pp. 101-124, 2000.

[126] Q. Gu, Z. Li and J. Han, “Generalized Fisher Score for Feature Selection,” in *UAI'11 Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, Barcelona, 2011.

# Appendix

## Appendix I- Ethical form- Approved

# EA1

[doc version 09.02]

UNIVERSITY OF LINCOLN

## Ethical Approval Form: Library/Desk/Lab/Studio-based Research Projects

This form must be completed for each piece of research activity whether conducted by academic staff, research staff, graduate students or undergraduates. Applications by students must be endorsed by an academic member of staff acting as Principal Investigator/supervisor. The completed form must be sent to the designated Ethics Committee within the College.

**Please complete all sections**. If a section is not applicable, write N/A.

| | | |
|---|---|---|
| 1 | **Name of Applicant** | Dalila Avdic |
| 2 | **School** | School of Engineering |
| 3 | **Position in the University** | PhD student |
| 4 | **Role in relation to this research** | Student researcher |
| 5 | **Name(s) of collaborators/co-workers and their relationship to the project (e.g. supervisor, assistant etc.)** | *Name, and role in project:*<br>1. Mike Gallimore- Supervisor 1<br>2.Mike Riley – Supervisor 2<br>3.Chris Bingham- Director of Studies |
| 6 | **Brief statement of main Research Question or Project Title** | Neutrophil count prediction for personalized drug dosing in childhood cancer patients receiving 6-mercaptopurine chemotherapy treatment |
| 7 | **Ethical checklist** | Does the research involve living human participants, or human tissue? **Yes** ☐ **No** ☒<br>*If you answered "yes", submit form EA2 for Ethical Approval.*<br>Does the research involve living animals, or animal tissue? **Yes** ☐ **No** ☒<br>*If you answered "yes", submit form EA3 for Ethical Approval.*<br>Does the research involve confidential data, or data not in the public domain? **Yes** ☒ **No** ☐<br>Does the project potentially put you or your collaborators at physical or psychological risk? **Yes** ☐ **No** ☒<br>Could the topic or results of this research be seen as illegal, or attract legal action against the University from an outside agency? **Yes** ☐ **No** ☒<br>Could the topic or results of this research attract unwelcome media attention, or affect the reputation or standing of the University? **Yes** ☐ **No** ☒<br>Could the topic, results or conduct of this research be regarded as offensive, immoral or destructive by some reasonable people? **Yes** ☐ **No** ☒<br>Does this research need to be undertaken under a relevant professional code of conduct? **Yes** ☐ **No** ☒<br>Are there any potential conflicts of interest in conducting this research, including financial gain for the researchers, or for individuals or external organizations affiliated with the researchers? **Yes** ☐ **No** ☒<br>Are there any factors inhibiting the application of the University's ethical guidelines, including those on proper treatment of data, |

| | |
|---|---|
| | research design and publication of results? Yes ☐ No ☒<br><br>Does the research require the approval of any external body? Yes ☐ No ☒<br><br>***If the answer to all questions above is "No", you may complete section 8 to certify that there are no ethical issues, submit this form to the relevant Ethics Committee, and proceed with the research immediately. You accept professional responsibility for this decision, and if unsure should instead submit to the Committee.***<br><br>***If the answer to any of the above questions is "Yes", complete the rest of the form, submit to the relevant Ethics Committee, and await approval before proceeding with the research. Answering "Yes" does not necessarily imply that the research is problematic, only the Ethics Committee needs to consider the research to ensure that it can proceed, and that the research design conforms to best practice.*** |
| **8 Self certification of Ethical Review** | *Having reviewed the ethical implications of this research, I certify that there are no issues requiring Ethical Approval. I certify that the research will be carried out in compliance with the University's ethical guidelines for library/desk/laboratory/studio-based research, with Health and Safety regulations, and with all other relevant University policies and procedures. If there are any changes to the research requiring ethical clearance, I shall apply for such clearance before continuing with the research.*<br><br>Signed:  <br><br>M GALLIMORE<br><br>Principal Investigator<br><br>***Note.*** *This section must be endorsed by the member of academic staff responsible for the project. In the case of research by students, the supervising member of academic staff must sign. The signed form should then be submitted to the relevant Ethics Committee within the College, and the research may proceed.* |
| **9 Does the research comply with the University's key ethical principles for library/desk/lab/studio-based research ?** | **Yes** ☒ **No** ☐<br><br>If "No", provide an ethical justification for your project and explain why you wish to continue with the research in breach of normal ethical principles:- |
| **10 If applicable, please state the relevant professional code(s) under which the research is being conducted and confirm compliance** | |
| **11 Does this research require the approval of an external body ?** | **Yes** ☐ **No** ☒<br><br>**If "Yes", please state which body:-** |

| | |
|---|---|
| 12 Has ethical approval already been obtained from that body ? | Yes ☐ -Please append documentary evidence to this form.<br>No ☐<br>If "No", please state why not:-<br><br>Please note that any such approvals must be obtained and documented before the project begins. |
| 13 If there are any other ethical issues, to which the attention of the approving committee should be drawn, please state them in this section, and explain how you have taken the issues into account, so that the research should be approved. Please consult the University's ethical guidelines for advice.<br><br>Please also include here, or attach separately, a brief description of the research, to allow the approving committee to reach judgement. | |

APPLICANT SIGNATURE

**I hereby request ethical approval for the research as described above.**
**I certify that I have read the University's ethical guidelines for library/desk/laboratory/studio-based research.**

10.08.2015

**Applicant Signature** **Date**

DALILA AVDIC

**PRINT NAME**

*FOR STUDENT APPLICATIONS ONLY –*
*Academic Support for Ethics*
Academic support should be sought prior to submitting this form to the designated Ethics Committee within the Faculty.

| | |
|---|---|
| • ***Undergraduate / Postgraduate Taught application*** | ***Academic Member of staff nominated by the School (consult your project tutor)*** |
| • ***Postgraduate Research Application*** | ***Director of Studies*** |

*I support the application for ethical approval*

Academic / Director of Studies Signature

13.08.2015
Date

**Professor Chris Bingham**
PRINT NAME

---

**FOR COMPLETION BY THE DESIGNATED ETHICS COMMITTEE WITHIN THE COLLEGE**

Please select ONE of A, B, C or D below:

☒ **A. Ethical approval to this research.**

☐ **B. Conditional ethical approval to this research.**

| | |
|---|---|
| **10 Please state the condition (inc. date by which condition must be satisfied if applicable)** | |

☐ **C. Ethical approval cannot be given to this research but the application is referred on to the University Research Ethics Committee for higher level consideration.**

| | |
|---|---|
| **11 Please state the reason** | |

☐ **D. Ethical approval cannot be given to this research and it is recommended that the research should not proceed.**

| | |
|---|---|
| **12 Please state the reason, bearing in mind the University's ethical framework, including the primary concern for Academic Freedom.** | |

**Signature of the Chair of the designated ethics committee within the College**

**Signature:** _____ **Date:** 7/9/15

Chair of College of Science Ethics Committee

**Key ethical guidelines for library/desk/laboratory/studio-based research**

The University of Lincoln has drawn up the following key principles for researchers engaged in library/desk/laboratory/studio-based projects in order to promote high professional standards. They should be read alongside the University's Ethical Principles for Conducting Research with Humans and Other Animals, and operate as part of the University's Ethical Framework.

- *Non-falsification of data*: Researchers have an ethical obligation to refrain from tampering with data. Thus questionnaire responses, experimental observations and data analyses should not be fabricated, altered nor discarded. In addition, researchers have a responsibility to exercise reasonable care in processing data to ensure no errors affect the results.
- *Ethics of reporting research*: Researchers are obliged to give full and proper attribution of ideas: presenting the words, data or ideas of another person as your own without properly citing them amounts to plagiarism. This is not only misconduct but can also be an infringement of copyright, amounting to theft of intellectual property.
- *Ethics and research design*: Researchers should be open to a range of methods: failure to consider and evaluate alternative methods and tools for the collection of data may be regarded as too overtly biased. All appropriate steps should be taken to ensure that no samples are obtained from unethical sources e.g. illegal databases; unregistered suppliers of samples from humans or other animals.
- *Authorship credit*: Only those researchers who are significant contributors to a research project should be given authorship credit. A "significant contributor" might be described as a person playing a major role in conceptualising, analysing or writing the final document. Ideally, all those involved in the research project should decide upon the order of authorship. Usually, the first author is the one who has made the biggest contribution.
- *Conflict of interest*: Researchers should be aware of the potential influence of personal or commercial interests on their work and take all practical measures to ensure that information is presented without distortion.
- *The principle of beneficence*: Researchers are required to protect individuals by seeking to maximise anticipated benefits and minimise possible harms. It is therefore necessary to examine carefully the design of the study and its risks and benefits including, in some cases, identifying alternative ways of obtaining the benefits sought from the research. Research risks must always be justified by the expected benefits of research.
- *Professional codes*: Researchers should undertake research legally and in accordance with any relevant professional codes of conduct.
- *Personal information*: Researchers should anonymise information which relates to individuals when they have not obtained informed consent, unless there is a clear justification to the contrary. They should also be aware of the impact of wider public dissemination of their work and the impact this might have on any individual or group of individuals. If it is anticipated that it might cause distress, it is essential to demonstrate that the benefits outweigh this risk.